ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ
(MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Παράλληλη προσομοίωση συστολικής συμπύκνωσης αραιών πινάκων και συμβολική αναπαράσταση συστολικών υπολογισμών στο χώρο»

Μπάρτζη Α. Αναστασία
Μ3950012

ΑΘΗΝΑ, ΟΚΤΩΒΡΙΟΣ 1997

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ
(MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Παράλληλη προσομοίωση συστολικής συμπύκνωσης αραιών πινάκων και συμβολική αναπαράσταση συστολικών υπολογισμών στο χώρο»

Μπάρτζη Α. Αναστασία

Μ3950012

Επιβλέπων Καθηγητής: Μιχάλης Π. Μπεκάκος

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΘΗΝΑ, ΟΚΤΩΒΡΙΟΣ 1997

Το αφιερώνω
στη
μητέρα μου Ευθυμία
και στην
εξαδέλφη μου Ιωάννα Γκάγκου

Πρόλογος

"Most of our thinking is actually done
unconsciously, and at such a fast speed
that it must be highly parallel - so we know and
have existence proofs that it can be done,
and that parallel algorithms do in fact exist"
Πηγή : Βιβλιογραφία
[67]

Τα συμβατικά von Neumann συστήματα υπολογιστών, με τη σημερινή εξέλιξη της τεχνολογίας, προσεγγίζουν ένα θεμελιώδες και φυσικό από τεχνολογικής άποψης όριο, αναφορικά με τις δυνατότητες περαιτέρω αύξησης των υπολογιστικών επιδόσεών τους. Η παράλληλη επεξεργασία αποτελεί έναν πολύ σπουδαίο και ραγδαία εξελισσόμενο χώρο ο οποίος προσφέρει τις δυνατότητες για απεριόριστη, θεωρητικά, υπολογιστική ισχύ.

Οι ερευνητικοί στόχοι στο χώρο της παράλληλης επεξεργασίας και των γλωσσών προγραμματισμού αφορούν στην ανεύρεση *αφαιρέσεων* (abstractions) οι οποίες θα γεφυρώνουν το χάσμα μεταξύ της ανθρώπινης σκέψης και της λειτουργίας των συστημάτων υπολογιστών. Τα τελευταία σαράντα χρόνια, η έρευνα για τέτοιου είδους αφαιρέσεις συνεχώς αντιμετώπιζε το δίλημμα της διευκόλυνσης του ανθρώπινου τρόπου σκέψης και έκφρασης, χωρίς να χάνεται η επαφή με την υπολογιστική βάση, την αρχιτεκτονική von Neumann.

Στην ύλη που ακολουθεί, παρουσιάζονται μερικά από τα βασικά χαρακτηριστικά των παράλληλων συστημάτων και των διαφόρων μορφών παράλληλης επεξεργασίας. Η μελέτη επικεντρώνεται ιδιαίτερα στα χαρακτηριστικά των αλγοριθμο-δομημένων συστολικών αρχιτεκτονικών και της συστολικής επεξεργασίας.

Πιο συγκεκριμένα, στο Κεφάλαιο 1, γίνεται αναφορά στις Αρχιτεκτονικές Συστημάτων Πολλαπλών Υπολογιστών, ενώ ταυτόχρονα παρουσιάζονται οι δυνατότητες και οι περιορισμοί της παράλληλης επεξεργασίας. Στο Κεφάλαιο 2, αναφέρονται οι βασικές μορφές παράλληλης επεξεργασίας, ενώ δίνεται ιδιαίτερη έμφαση στη συστολική και στην κυματοειδούς μορφής επεξεργασία. Στο Κεφάλαιο 3, παρουσιάζεται μία συγκριτική

μελέτη διαφόρων προσεγγίσεων που αφορούν σε, ευρετικές και μη, τεχνικές ελαχιστοποίησης του εύρους ζώνης και της κατατομής αραιών πινάκων. Η συνοπτική αυτή συγκριτική παρουσίαση ολοκληρώνεται στο Κεφάλαιο 4, στο οποίο παρουσιάζεται μία δυναμική συστολική προσέγγιση, η οποία βασίζεται και ταυτόχρονα συγκρίνεται με δύο ευρέως χρησιμοποιούμενους αλγόριθμους, τον αλγόριθμο των Cuthill-Mckee και τον αλγόριθμο των Gibbs, et al.

Στα Κεφάλαια 5 και 6, κάνοντας χρήση εννοιών της Προβολικής Γεωμετρίας, εισάγεται και μελετάται, ένα δυναμικό συμβολο-εργαλείο περιγραφής (Symbolic Descriptive Tool -SDT) των συστολικών αρχιτεκτονικών και της υπολογιστικής δραστηριότητας των αντίστοιχων αλγορίθμων, το οποίο χρησιμοποιείται για τη δημιουργία βέλτιστων αλγοριθμο-δομημένων επιπεδικών συστολικών τοπολογιών για την επίλυσή τους. Δύο χαρακτηριστικοί ως προς την απόδοσή τους αλγόριθμοι χρησιμοποιούνται για την παρουσίαση της πρακτικής χρησιμότητας του εργαλείου SDT, το οποίο αποσκοπεί στην περαιτέρω απλοποίηση των συστολικών αλγορίθμων μέσω της επιλογής του καταλληλότερου δυνατού δικτύου φυσικών επεξεργαστών στα οποία θα επιλύονται.

Η Ελληνική Ορολογία Πληροφορικής που χρησιμοποιείται έχει εκπονηθεί από την Ομάδα Εργασίας ΟΕ-1 "*Ορολογία Πληροφορικής*", της Τεχνικής Επιτροπής ΤΕ-48 *"Ηλεκτρονική Επεξεργασία Πληροφοριών"*, που εκπονεί Ελληνικά Πρότυπα (ΕΛΟΤ) με βάση τα Διεθνή Πρότυπα ISO. Παρόλα αυτά, για την αποφυγή οποιωνδήποτε παρερμηνειών, η εκάστοτε ελληνική απόδοση συνοδεύεται από την αντίστοιχη Αγγλική.

Επιθυμώ να ευχαριστήσω τον επιβλέποντα την εργασία μου *Αναπληρωτή Καθηγητή κ. Μιχάλη Π. Μπεκάκο* για το ενδιαφέρον και την ανεκτίμητη βοήθειά του για την ολοκλήρωση των ερευνητικών προσπαθειών μου και τη συγγραφή της παρούσας εργασίας. Τέλος, επιθυμώ ιδιαίτερα να ευχαριστήσω για περισσότερα από όσα μπορούν να εκφραστούν με λόγια την *εξαδέλφη μου Ιωάννα Γκάγκου* και το *θείο μου Γεώργιο Γκάγκο* για την αμέριστη ηθική και υλική συμπαράστασή τους κατά τη διάρκεια των σπουδών μου.

Αθήνα, Οκτώβριος 1997.

Αναστασία Α. Μπάρτζη

Περιεχόμενα

Κεφάλαιο 1

Συμπλέγματα Υπολογιστών Υψηλής Απόδοσης

Κεφάλαιο 2

Μορφές Παράλληλης Επεξεργασίας και Αλγοριθμο-δομημένες Αρχιτεκτονικές

Κεφάλαιο 3

Σχήματα Συμπύκνωσης Αραιών Πινάκων

Κεφάλαιο 4

Ελαχιστοποίηση του Εύρους Ζώνης Αραιών Πινάκων σε Συστολικές Αρχιτεκτονικές

Κεφάλαιο 5

Συμβολικό Περιγραφικό Εργαλείο Απόδοσης Συστολικών Αλγορίθμων

Κεφάλαιο 6

Συμβολική Αναπαράσταση Συστολικών Αλγορίθμων

Κεφάλαιο 1

Συμπλέγματα Υπολογιστών Υψηλής Απόδοσης

1.1 Εισαγωγή

Γενικά, ο νόμος της φύσης χαρακτηρίζεται από μία αλληλουχία γεγονότων και φαινομένων, τα οποία επαναλαμβάνονται περιοδικά σε τακτά χρονικά διαστήματα. Ειδικότερα, κάθε ανθρώπινη δραστηριότητα όπως, ο συνειδητός τρόπος σκέψης, λόγος και η έκφραση της γνώσης, αλλά ακόμη και αυτή η θεμελιώδης έννοια του χρόνου, βασίζονται σ' αυτή ακριβώς την αρχή της ακολουθιακής (sequential) διαδοχής σκέψεων, λέξεων και διακριτών χρονικών μονάδων. Ακόμη και πριν από την εμφάνιση των πρώτων ηλεκτρονικών υπολογιστών, η μαθηματική έκφραση ενός αλγόριθμου αποδιδόταν ως μία πεπερασμένη ακολουθία διακεκριμένων λειτουργιών. Κατά συνέπεια, ήταν απόλυτα φυσικό να βασιστούν σε μία σειριακή μορφή επεξαργασίας, τόσο η λειτουργία των πρώτων Η/Υ που κατασκευάστηκαν, όσο και η διαμόρφωση των πρώτων προγραμμάτων που αναπτύχθηκαν.

Για περίπου 40 και πλέον χρόνια από την εμφάνιση του πρώτου ηλεκτρονικού υπολογιστή ΕΝΙΑC το 1945, οι προσπάθειες επικεντρώθηκαν κυρίως στην ανάπτυξη σειριακών προγραμμάτων για την επίλυση των διαφόρων προβλημάτων που απασχολούσαν την επιστημονική κοινότητα. Η συνεχής και ταυτόχρονα ραγδαία αύξηση της υπολογιστικής ισχύος των Η/Υ (οι σημερινοί σειριακοί

επεξεργαστές είναι τουλάχιστον τέσσερις τάξεις μεγέθους ταχύτεροι από τον ENIAC), η οποία οφειλόταν, κυρίως, στη σταθερή και συνεχή τεχνολογική πρόοδο και εξέλιξη, τους καθιέρωσε πολύ γρήγορα και έκανε πολλούς ανθρώπους να πιστεύουν ότι ένας σειριακός υπολογιστής, ο οποίος διαθέτει έναν γενικού σκοπού (general purpose) επεξεργαστή, είναι υπεραρκετός για να καλύψει τις εκάστοτε απαιτήσεις, ενώ στο βαθμό που κάτι τέτοιο δεν ισχύει άμεσα θα ισχύσει, αναμφίβολα, στο άμεσο μέλλον. Ωστόσο, οι παραδοσιακοί σειριακοί υπολογιστές σύντομα θα προσεγγίσουν το ανώτατο όριό τους, το λεγόμενο *'φράγμα της ταχύτητας του φωτός'*, όσον αφορά στην παρεχόμενη ταχύτητα και υπολογιστική ισχύ. Ακόμη και τότε, όμως, η ταχύτητα των παραδοσιακών σειριακών υπολογιστών δε θα επαρκεί για το χειρισμό πολλών και εξαιρετικά σημαντικών προβλημάτων, όπως για παράδειγμα είναι η επεξεργασία εικόνων, η πρόβλεψη του καιρού και των σεισμών, η μοντελοποίηση του ανθρώπινου εγκεφάλου και η ανάπτυξη νοήμονων προγραμμάτων, κ.ά.

Καθώς, λοιπόν, ο χώρος της επιστήμης των υπολογιστών εξελισσόταν βαθμιαία, γινόταν ολοένα και περισσότερο αντιληπτό το γεγονός ότι η σειριακή μορφή επεξεργασίας από μόνη της δεν επαρκούσε πλέον. Το γεγονός αυτό μπορεί να επιβεβαιωθεί και από την ανάγκη, σε οποιονδήποτε τομέα, για καταμερισμό του αντίστοιχου έργου, με σκοπό την επίτευξη της βέλτιστης επίδοσης και απόδοσης, μέσα από τη συντονισμένη και παράλληλη δράση πολλών μεμονωμένων ατόμων. Θα πρέπει να σημειωθεί, βέβαια, ότι η σειριακή και η παράλληλη μορφή επεξεργασίας δεν υφίστανται ως αμοιβαία αποκλειόμενες έννοιες, αλλά αντίθετα απαιτείται να συνυπάρχουν και να λειτουργούν συμπληρωματικά.

Με δεδομένο, λοιπόν, το σκοπό του σχεδιασμού και της κατασκευής των Η/Υ, το βασικό πρόβλημα αφορά στην ανακάλυψη του καλύτερου δυνατού τρόπου χρήσης και αξιοποίησης αυτού του ευέλικτου και δυναμικού εργαλείου, το οποίο θεωρητικά διαθέτει απεριόριστες δυνατότητες. Έτσι, μολονότι δεν είναι δυνατό να κατασκευάζονται επ' αόριστον ταχύτεροι σειριακοί υπολογιστές, είναι δυνατόν να κατασκευάζονται ολοένα και μεγαλύτερα συμπλέγματα παράλληλων-σειριακών υπολογιστών, αυξάνοντας το πλήθος των επεξεργαστών και κατά συνέπεια, το δυναμικό της προκύπτουσας αρχιτεκτονικής.

Κάτι τέτοιο, αφενός μεν αυξάνει κατακόρυφα την υπολογιστική ισχύ των Η/Υ, αφετέρου, στις περισσότερες περιπτώσεις, προυποθέτει τον εκ νέου σχεδιασμό των

αλγορίθμων για την πλήρη εκμετάλλευση όχι μόνο των ανεξάρτητων ενοτήτων που εντοπίστηκαν στη σειριακή σχεδίαση, αλλά πιθανόν και άλλου ενυπάρχοντος παράλληλου δυναμικού το οποίο δεν μπορεί να εκδηλωθεί σε μία σειριακής μορφής αλγοριθμική ανάπτυξη. Είναι σχεδόν βέβαιο, ότι σε 50 περίπου χρόνια από σήμερα, οι άνθρωποι θα κοιτάζουν πίσω και θα αναρωτιούνται γιατί παρόλη την απεριόριστη υπολογιστική ισχύ των Η/Υ, η ουσιαστική χρήση τους περιορίζεται σε ένα σύνολο απλών σχετικά προβλημάτων. Το γεγονός αυτό οφείλεται, κατεξοχήν, στο ότι πολλά από τα προβλήματα στην επίλυση των οποίων θα εστιάσουν οι υπολογιστές του 21ου αιώνα, είτε δεν έχουν ακόμη συλληφθεί ως ιδέα ή στην καλύτερη περίπτωση δεν έχουν διαμορφωθεί κατάλληλα ώστε να μπορούν να υποστούν επεξεργασία από έναν Η/Υ, λόγω της υψηλού επιπέδου πολυπλοκότητας που επιδεικνύουν, η οποία δυσχεραίνει σημαντικά το σχεδιασμό των αντίστοιχων αλγόριθμων.

Αποφασιστικό ρόλο για την επίτευξη του παραπάνω στόχου αναμένεται να παίξει η δυνατότητα της άμεσης έκφρασης των αλγορίθμων υπό μορφή γραφημάτων, τα οποία θα προσδιορίζουν την πλέον κατάλληλη δομή επεξεργαστών για την επίλυση του εκάστοτε προβλήματος. Κατόπιν θα επιλέγεται, αναδιαρθρώνεται ή κατασκευάζεται το αντίστοιχο δίκτυο φυσικών επεξεργαστών, το οποίο θα χρησιμοποιείται για την πραγματική εκτέλεση του αλγόριθμου. Στην καλύτερη περίπτωση, οι σημερινοί αλγόριθμοι θα είναι δυνατό να εκφραστούν με διαγράμματα ροής (flow diagrams), τα οποία θα προσδιορίζουν ισομορφικά (isomorphic) δίκτυα επεξεργαστών, ικανά να εκτελέσουν τους αλγόριθμους αυτούς.

1.2 Παράλληλα Συστήματα

Παρά τη ραγδαία αύξηση της υπολογιστικής ισχύος των επεξεργαστών Πολύ Υψηλής Κλίμακας Ολοκλήρωσης (Very Large Scale Integration - VLSI), η αναγκαιότητα της χρήσης παράλληλων υπολογιστών είναι πλέον δεδομένη, εφόσον όπως ήδη αναφέρθηκε, υπάρχει ένα ευρύ φάσμα σημαντικών προβλημάτων τα οποία δεν είναι δυνατό να επιλυθούν με κανενός είδους σειριακό υπολογιστή. Κάτι τέτοιο είναι ιδιαίτερα εμφανές σε περιπτώσεις στις οποίες απαιτείται η επίλυση προβλημάτων σε *πραγματικό χρόνο*, πράγμα που σημαίνει ότι ο εκάστοτε χρησιμοποιούμενος υπολογιστής θα πρέπει να είναι αρκετά γρήγορος ώστε να

μπορεί να αλληλεπιδρά άμεσα με τον εξωτερικό κόσμο, λαμβάνοντας υπόψη όλες τις πραγματοποιούμενες μεταβολές που συμβάλλουν στη λήψη των βέλτιστων δυνατών αποτελεσμάτων. Τέτοιου είδους προβλήματα, καθώς επίσης και πολλά άλλα, δεν είναι δυνατό να επιλυθούν εξαιτίας του μεγέθους της υπολογιστικής ισχύος που απαιτούν.

Μία προσέγγιση για την επίτευξη της απαιτούμενης υπολογιστικής ισχύος αφορά στον εκ νέου σχεδιασμό ταχύτερων σειριακών επεξεργαστών μέσω της συγχώνευσης μεγάλου πλήθους VLSI τσιπς. Ωστόσο, ο σχεδιασμός και η κατασκευή τέτοιου είδους εξειδικευμένων επεξεργαστών έχει αποδειχθεί ότι είναι εξαιρετικά δαπανηρή. Πρίν από 20 περίπου χρόνια, οι κατασκευαστές Η/Υ άρχισαν να ανακαλύπτουν ότι η ενσωμάτωση πολλών τυπικών VLSI επεξεργαστών σε έναν και μόνο υπολογιστή, απαιτούσε πολύ λιγότερο οικονομικό κόστος σε σχέση με τον εκ νέου σχεδιασμό ενός ταχύτερου επεξεργαστή. Έτσι, στο βαθμό που απαιτείτο δεκαπλάσια υπολογιστική ισχύς χρησιμοποιούνταν 10 επεξεργαστές (τουλάχιστον), για εκατονταπλάσια υπολογιστική ισχύ, 100 επεξεργαστές (τουλάχιστον), κ.ο.κ. Η εφαρμογή αυτής ακριβώς της ιδέας εγκαινίασε τη δυναμική είσοδο των συστημάτων πολλαπλών επεξεργαστών (multiple processor systems) στο χώρο της αγοράς των υπολογιστών.

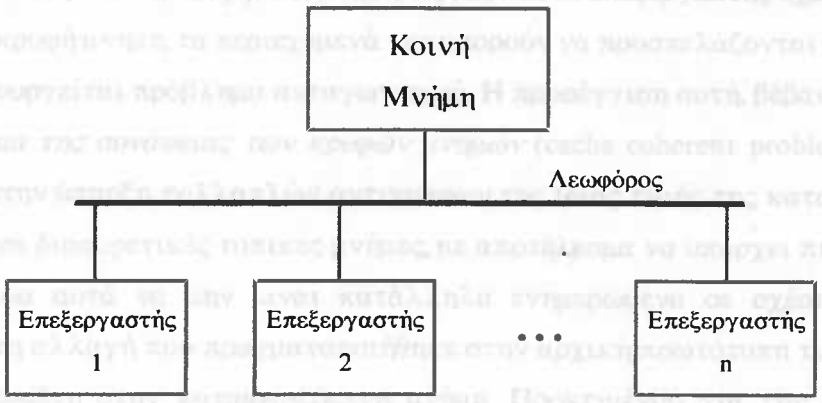
Το βασικό πλεονέκτημα της παραπάνω ιδέας ήταν και εξακολουθεί να είναι η δυνατότητα αύξησης της υπολογιστικής ισχύος σε οποιοδήποτε θεωρητικά επιθυμητό επίπεδο, εφόσον κάθε νέα γενιά ταχύτερων VLSI σειριακών επεξεργαστών που αναπτύσσεται είναι δυνατό να χρησιμοποιηθεί για την κατασκευή ταχύτερων συστημάτων πολλαπλών επεξεργαστών. Το γεγονός αυτό επιβεβαιώνει και τη συμπόρευση της παράλληλης και της σειριακής μορφής επεξεργασίας, οι οποίες συνλειτουργούν στο πλαίσιο ενός και μόνο υπολογιστικού συστήματος. Κατά συνέπεια, οποιαδήποτε πρόοδος που αφορά στο σχεδιασμό και στην κατασκευή ταχύτερων VLSI επεξεργαστών συμβάλλει στη δημιουργία των βασικών δομοστοιχείων ταχύτερων και περισσότερο ισχυρών παράλληλων υπολογιστών. Ακόμη και τα σύνολα των τσιπς που χρησιμοποιούνται στη διανυσματική μορφή επεξεργασίας και που από μόνα τους παρέχουν υψηλό επίπεδο παραλληλισμού, είναι δυνατόν να συνδυαστούν περαιτέρω σε συστήματα πολλαπλών επεξεργαστών προκειμένου να επιτευχθεί ακόμη μεγαλύτερος

παραλληλισμός. Το βέβαιο, πάντως, είναι ότι η τεχνολογική πρόοδος θα συνεχίσει να ενισχύει την πρόοδο των παράλληλων συστημάτων πολλαπλών επεξεργαστών.

Η ύπαρξη, όμως, πολλαπλών επεξεργαστών σε έναν παράλληλο υπολογιστή, συνεπάγεται κάποιες επιπλέον απαιτήσεις όσον αφορά στην αρχιτεκτονική οργάνωση αυτού του υπολογιστή. Βασική προυπόθεση, λοιπόν, για την επίτευξη της ταυτόχρονης επεξεργασίας δεδομένων του ίδιου προβλήματος από περισσότερους του ενός επεξεργαστές, αποτελεί η από κοινού δυνατότητα πρόσβασης στο ίδιο σύνολο δεδομένων και η αμοιβαία επικοινωνία μεταξύ τους. Δύο είναι οι βασικές αρχιτεκτονικές προσεγγίσεις οι οποίες ικανοποιούν την παραπάνω απαίτηση : Η αρχιτεκτονική καταμεριζόμενης μνήμης (shared memory) και η αρχιτεκτονική διαβίβασης μηνυμάτων (message passing). Καθένας από τους δύο αυτούς τύπους αρχιτεκτονικών παράλληλων υπολογιστών παρουσιάζει συγκεκριμένα πλεονεκτήματα, αλλά και μειονεκτήματα, στα οποία θα αναφερθούμε συνοπτικά στη συνέχεια του παρόντος Κεφαλαίου.

1.3 Αρχιτεκτονική Συστημάτων Πολυεπεξεργασίας

Το βασικό χαρακτηριστικό των υπολογιστών καταμεριζόμενης μνήμης, οι οποίοι συνήθως καλούνται συστήματα *πολυεπεξεργασίας*, είναι ότι κάθε μεμονωμένος επεξεργαστής έχει τη δυνατότητα πρόσβασης σε μία διαμοιραζόμενη περιοχή μνήμης, γεγονός που επιτρέπει την από κοινού χρήση διαφόρων μεταβλητών και δομών δεδομένων που είναι αποθηκευμένες σε αυτή. Μία τυπική οργάνωση ενός τέτοιου συστήματος, το οποίο διαθέτει μία κοινή λεωφόρο (common bus) απεικονίζεται στο σχήμα 1.3-1. Όλοι οι επεξεργαστές λειτουργούν παράλληλα και συνδέονται με την κοινή λεωφόρο, ενώ καθένας από αυτούς μπορεί να προσπελάζει την κοινή μνήμη μέσω της λεωφόρου αυτής. Η κοινή λεωφόρος είναι υπεύθυνη όχι μόνον για τη διευθέτηση των ταυτόχρονων αιτήσεων στη μνήμη από πολλαπλούς επεξεργαστές, αλλά και για την εξασφάλιση της δίκαιης εξυπηρέτησης, από την άποψη της καθυστέρησης στην προσπέλαση των επεξεργαστών. Η δραστηριότητα κάθε επεξεργαστή είναι παρόμοια με τη δραστηριότητα του επεξεργαστή ενός συνηθισμένου σειριακού υπολογιστή και περιλαμβάνει το διάβασμα των τιμών των δεδομένων από την κοινή μνήμη, τον υπολογισμό των νέων τιμών και την επανεγγραφή αυτών στην κοινή μνήμη.



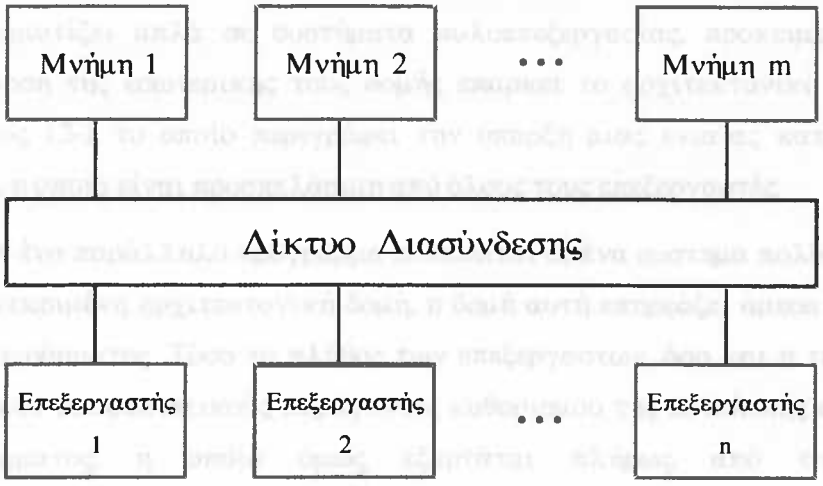
Σχήμα 1.3-1: Οργάνωση Συστήματος Πολυεπεξεργασίας με Κοινή Λεωφόρο

Η διαφορά έγκειται στο ότι την ίδια, πλέον, εργασία πραγματοποιούν παράλληλα n επεξεργαστές, αντί για έναν, και κατά συνέπεια, η μέγιστη υπολογιστική ισχύς είναι n φορές μεγαλύτερη σε σχέση με αυτήν του σειριακού υπολογιστή. Το πλήθος των επεξεργαστών οι οποίοι είναι διαθέσιμοι σε τέτοιου είδους συστήματα ανέρχεται σήμερα, το πολύ, σε 32 επεξεργαστές εξαιτίας του ότι μεγαλύτερο πλήθος επεξεργαστών μειώνει σημαντικά την απόδοση του συστήματος. Ένα από τα βασικότερα προβλήματα των συστημάτων αυτής της κατηγορίας είναι ο ανταγωνισμός (contention) των επεξεργαστών για την προσπέλαση της κοινής μνήμης, πρόβλημα το οποίο μεγενθύνεται καθώς αυξάνονται οι επεξεργαστές. Το πρόβλημα αυτό εμφανίζεται όταν πολλοί επεξεργαστές προσπαθούν να προσπελάσουν τη μνήμη κατά τη διάρκεια του ίδιου χρονικού διαστήματος, με αποτέλεσμα να μην είναι δυνατόν να εξυπηρετηθούν όλες οι αιτήσεις ταυτόχρονα. Συνήθως, άλλες από αυτές εξυπηρετούνται άμεσα, ενώ οι υπόλοιπες αναστέλλονται μέχρις ότου καταστεί εφικτή η εξυπηρέτησή τους. Πολλοί επιστήμονες και ερευνητές έχουν επικεντρώσει τις προσπάθειές τους στην ανεύρεση τρόπων για τη μείωση της πιθανότητας εμφάνισης ανταγωνισμού για την προσπέλαση της μνήμης.

Για το σκοπό αυτό, έχει αναπτυχθεί ένα πλήθος διαφορετικών τεχνικών, οι οποίες βοηθούν στη μείωση του ανταγωνισμού συμβάλλοντας, έτσι, στην αύξηση της αποδοτικότητας του συστήματος. Μία τέτοια τεχνική είναι να μπορεί κάθε επεξεργαστής να διαθέτει μία κρυφή μνήμη (cache memory), η οποία θα χρησιμοποιείται για να κρατάει αντίγραφα των πιο πρόσφατα χρησιμοποιούμενων

τιμών από την καταμεριζόμενη μνήμη. Εφόσον κάθε επεξεργαστής έχει τη δική του τοπική κρυφή μνήμη, τα περιεχόμενά της μπορούν να προσπελάζονται άμεσα, χωρίς να δημιουργείται πρόβλημα ανταγωνισμού. Η προσέγγιση αυτή, βέβαια, εισάγει το *πρόβλημα της συνάφειας των κρυφών μνημών* (cache coherent problem) το οποίο αφορά στην ύπαρξη πολλαπλών αντιγράφων της ίδιας τιμής της καταμεριζόμενης μνήμης σε διαφορετικές τοπικές μνήμες, με αποτέλεσμα να υπάρχει πιθανότητα τα αντίγραφα αυτά να μην είναι κατάλληλα ενημερωμένα σε σχέση με κάποια πρόσφατη αλλαγή που πραγματοποιήθηκε στην αρχική πρωτότυπη τιμή που είναι αποθηκευμένη στην καταμεριζόμενη μνήμη. Προκειμένου για την επίλυση του τελευταίου αυτού προβλήματος έχουν αναπτυχθεί περισσότερο εξεζητημένες τεχνικές, οι οποίες ποικίλλουν από σύστημα σε σύστημα. Μία από αυτές προτείνει τη χρήση κρυφών μνημών - *'κατασκόπων'* (snooping cache), οι οποίες παρακολουθούν συνεχώς την κοινή λεωφόρο και ενημερώνουν άμεσα τα περιεχόμενά τους, ανάλογα με τις τροποποιήσεις που πραγματοποιούν άλλοι επεξεργαστές στα περιεχόμενα της καταμεριζόμενης μνήμης.

Μία διαφορετική τεχνική, η οποία χρησιμοποιείται επίσης για τη μείωση του ανταγωνισμού για την κατοχή της καταμεριζόμενης μνήμης, αφορά στη διαίρεση της μνήμης αυτής σε ξεχωριστά δομοστοιχεία (modules), τα οποία μπορούν να προσπελαστούν παράλληλα από διαφορετικούς επεξεργαστές. Τα δεδομένα αποθηκεύονται σε πολλά διαφορετικά δομοστοιχεία μνήμης και έτσι μειώνεται σημαντικά η πιθανότητα εμφάνισης ανταγωνισμού σε καθένα από αυτά. Στο σχήμα 1.3-2 παρουσιάζεται η γενική οργάνωση ενός συστήματος πολυεπεξεργασίας με πολλαπλά δομοστοιχεία μνήμης. Καθένας από τους n επεξεργαστές μπορεί να προσπελάσει οποιοδήποτε από τα m δομοστοιχεία μνήμης διαμέσω ενός δικτύου διασύνδεσης επεξεργαστών και μνημών (processor-memory connection network). Το δίκτυο διασύνδεσης παρέχει ένα συγκεκριμένο βαθμό παραλληλισμού εξασφαλίζοντας ότι περισσότεροι του ενός επεξεργαστές μπορούν να προσπελάζουν παράλληλα διαφορετικά δομοστοιχεία μνήμης. Στην περίπτωση αυτή, το κόστος και η απόδοση του συστήματος εξαρτώνται από τα σχεδιαστικά χαρακτηριστικά του δικτύου διασύνδεσης. Οι περισσότερο διαδεδομένες σχεδιάσεις τέτοιων δικτύων είναι οι σχεδιάσεις *πεταλούδας* (butterfly), *ανάμειξης-ανταλλαγής* (shuffle-exchange), η *διαραβδωτική* (crossbar) και η *ωμέγα* (omega).



Σχήμα 1.3-2 Οργάνωση Συστήματος Πολυεπεξεργασίας με Πολλαπλά Δομοστοιχεία Μνήμης

Αναμφίβολα, η ύπαρξη πολλαπλών δομοστοιχείων μνημών συντελεί στη βελτίωση της απόδοσης του συστήματος, εφόσον, όπως ήδη αναφέρθηκε, παρέχεται η δυνατότητα παράλληλης προσπέλασής τους. Ταυτόχρονα, ο παραλληλισμός αυτός, σε επίπεδο προσπέλασης δομοστοιχείων μνήμης, ενισχύει την αύξηση του παραλληλισμού σε επίπεδο δραστηριότητας επεξεργαστών, εφόσον αυτοί δεν αναλώνονται τόσο πολύ πλέον στην αναμονή για την κατοχή της μνήμης. Επιπλέον, ένα βασικό χαρακτηριστικό όλων των συστημάτων που χρησιμοποιούν αυτή την οργάνωση είναι ότι από λογική άποψη η μνήμη θεωρείται ως ενιαία, ακόμη και όταν αυτή υποδιαιρείται σε δομοστοιχεία. Η δραστηριότητα του δικτύου διασύνδεσης καθορίζεται πλήρως σε επίπεδο υλικού, ενώ δεν είναι ορατή στον απλό προγραμματιστή ο οποίος αντιλαμβάνεται μία ενιαία καταμεριζόμενη μνήμη. Κατά συνέπεια, το προγραμματιστικό μοντέλο δε διαφοροποιείται μεταξύ των δύο οργανώσεων που απεικονίζονται στα σχήματα 1.3-1 και 1.3-2. Και στις δύο περιπτώσεις θεωρείται ότι υπάρχει ένας μόνον χώρος διευθύνσεων μνήμης, τον οποίο όλοι οι επεξεργαστές αντιλαμβάνονται με τον ίδιο ακριβώς τρόπο. Στη δεύτερη οργάνωση, οι διευθύνσεις αυτές εκτείνονται, ουσιαστικά, κατά μήκος των πολλαπλών δομοστοιχείων μνημών, αλλά αυτό δεν είναι εμφανές στους επεξεργαστές. Όταν ένας επεξεργαστής διαβάζει ή γράφει σε μία συγκεκριμένη διεύθυνση μνήμης, η δραστηριότητα του δικτύου διασύνδεσης και η επιλογή του κατάλληλου δομοστοιχείου μνήμης καθορίζονται αυτόματα από το μηχανισμό

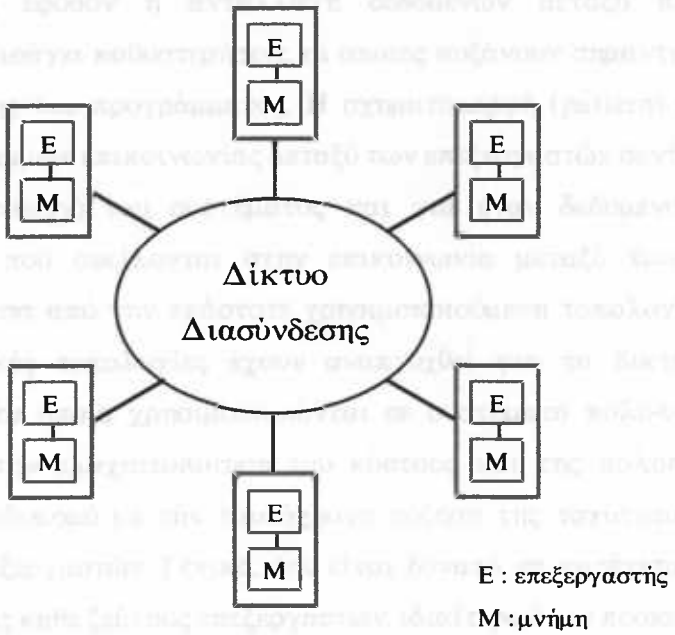
προσπέλασης της μνήμης που υπάρχει στο υλικό. Κατά συνέπεια, για κάποιον που προγραμματίζει απλά σε συστήματα πολυεπεξεργασίας, προκειμένου για την κατανόηση της εσωτερικής τους δομής επαρκεί το αρχιτεκτονικό μοντέλο του σχήματος 1.3-1, το οποίο περιγράφει την ύπαρξη μιας ενιαίας καταμεριζόμενης μνήμης, η οποία είναι προσπελάσιμη από όλους τους επεξεργαστές.

Όταν ένα παράλληλο πρόγραμμα εκτελείται σε ένα σύστημα πολυεπεξεργασίας με συγκεκριμένη αρχιτεκτονική δομή, η δομή αυτή επηρεάζει άμεσα την απόδοση του προγράμματος. Τόσο το πλήθος των επεξεργαστών, όσο και η ταχύτητά τους αποτελούν αποφασιστικούς παράγοντες καθορισμού της συνολικής απόδοσης του προγράμματος, η οποία όμως εξαρτάται πλήρως από την εκάστοτε χρησιμοποιούμενη οργάνωση της διασύνδεσης μεταξύ επεξεργαστών και μνημών. Οποιαδήποτε οργάνωση, όμως, κι αν χρησιμοποιηθεί σε αυτή την κατηγορία συστημάτων, πάντοτε υπάρχει η πιθανότητα εμφάνισης σημαντικά μειωμένης απόδοσης εξαιτίας της ύπαρξης ανταγωνισμού μεταξύ των επεξεργαστών για την κατοχή της καταμεριζόμενης μνήμης. Σε ορισμένες περιπτώσεις, η ίδια η φύση του προβλήματος είναι εκείνη που επιβάλλει έναν τέτοιο ανταγωνισμό, ο οποίος αναπόφευκτα εκδηλώνεται στο αντίστοιχο σύστημα υλοποίησης. Άλλες πάλι φορές, ο εκάστοτε παράλληλος αλγόριθμος επιτρέπει την εμφάνιση ανταγωνισμού μόνον σε συστήματα πολυεπεξεργασίας με συγκεκριμένη αρχιτεκτονική δομή.

1.4 Αρχιτεκτονική Συστημάτων Πολυ-υπολογιστών

Ένα σύστημα πολυ-υπολογιστών κατανεμημένης μνήμης (distributed memory) αποτελείται από πολλούς επεξεργαστές, καθώς επίσης και από ένα δίκτυο διασύνδεσης (interconnection network) για τη μεταξύ των επεξεργαστών επικοινωνία μέσω της διαβίβασης μηνυμάτων. Καθένας από τους επεξεργαστές είναι ένας αυτόνομος υπολογιστής, ο οποίος διαθέτει έναν επεξεργαστή, τοπική μνήμη και μερικές φορές ακόμη και ενσωματωμένους δίσκους ή περιφερειακές συσκευές εισόδου/εξόδου, ενώ μπορεί να πραγματοποιήσει τους υπολογισμούς του ανεξάρτητα από τους υπόλοιπους επεξεργαστές κάνοντας χρήση των δεδομένων που είναι αποθηκευμένα στην τοπική του μνήμη. Επιπλέον, κάθε επεξεργαστής μπορεί να στείλει και να παραλάβει δεδομένα από οποιονδήποτε άλλο επεξεργαστή, χρησιμοποιώντας το δίκτυο διασύνδεσης, το οποίο παρέχει στατικές

συνδέσεις από σημείο-σε-σημείο (point-to-point) μεταξύ των επεξεργαστών. Όλες οι διαθέσιμες τοπικές μνήμες είναι ιδιωτικές και μπορούν να προσπελαστούν μόνον από τους αντίστοιχους τοπικούς επεξεργαστές. Για το λόγο αυτό, τα παραδοσιακά συστήματα πολυ-υπολογιστών είναι γνωστά και ως μηχανές *μη απομακρυσμένης προσπέλασης της μνήμης* (no-remote-memory-access - NORMA). Ωστόσο, ο περιορισμός αυτός βαθμιαία εξαλείφεται και τα μελλοντικά συστήματα πολυ-υπολογιστών θα διαθέτουν κατανεμημένες-καταμεριζόμενες (distributed-shared) μνήμες. Το γενικό μοντέλο αρχιτεκτονικής ενός συστήματος πολυ-υπολογιστών περιγράφεται στο σχήμα 1.4-1.



Σχήμα 1.4-1: Οργάνωση Συστήματος Πολυ-υπολογιστή

Η βασική οργάνωση των συστημάτων πολυ-υπολογιστών παρουσιάζει ριζικές διαφορές σε σχέση με αυτή των συστημάτων πολυεπεξεργασίας και για το λόγο αυτό απαιτείται η χρήση διαφορετικών προγραμματιστικών μοντέλων για την ανάπτυξη παράλληλων προγραμμάτων. Στα συστήματα αυτής της κατηγορίας κάθε επεξεργαστής γνωρίζει τη μεταξύ τοπικής και απομακρυσμένης μνήμης. Η διαφορά αυτή έγκειται στη δυνατότητα άμεσης προσπέλασης της τοπικής μνήμης και έμμεσης προσπέλασης της απομακρυσμένης μνήμης, μέσω της ανταλλαγής δεδομένων με άλλους επεξεργαστές κάνοντας χρήση του δικτύου διασύνδεσης. Το γεγονός αυτό επιτρέπει την ελεύθερη πρόσβαση όλων των επεξεργαστών σε όλα τα

δεδομένα, στο βαθμό που κάτι τέτοιο είναι επιθυμητό. Κατά συνέπεια, η κατάργηση της ενιαίας μνήμης και η καθιέρωση μόνον τοπικών ιδιωτικών μνημών εξαλείφει πλήρως το πρόβλημα του ανταγωνισμού για την προσπέλαση της καταμεριζόμενης μνήμης.

Όταν ένας επεξεργαστής δε διαθέτει απευθείας γραμμές επικοινωνίας προς κάποιον άλλο επεξεργαστή, η επικοινωνία μεταξύ τους εξακολουθεί να είναι εφικτή και πραγματοποιείται με την προώθηση των δεδομένων μέσω των επεξεργαστών που βρίσκονται στη διαδρομή που συνδέει τον αποστολέα με τον παραλήπτη. Το γεγονός αυτό συχνά αποτελεί το βασικότερο παράγοντα μείωσης της απόδοσης, εφόσον η ανταλλαγή δεδομένων μεταξύ απομακρυσμένων επεξεργαστών εισάγει καθυστερήσεις, οι οποίες αυξάνουν σημαντικά το συνολικό χρόνο εκτέλεσης του προγράμματος. Η σχηματομορφή (pattern) αναπαράστασης των άμεσων γραμμών επικοινωνίας μεταξύ των επεξεργαστών συνήθως ονομάζεται *τοπολογία* (topology) του συστήματος και για έναν δεδομένο αλγόριθμο οι καθυστερήσεις που οφείλονται στην επικοινωνία μεταξύ των επεξεργαστών εξαρτώνται άμεσα από την εκάστοτε χρησιμοποιούμενη τοπολογία. Ένα σύνολο από διαφορετικές τοπολογίες έχουν αναπτυχθεί για τα δίκτυα διασύνδεσης επεξεργαστών τα οποία χρησιμοποιούνται σε συστήματα πολυ-υπολογιστών, με βασικό στόχο την ελαχιστοποίηση του κόστους και της πολυπλοκότητας του δικτύου, σε συνδυασμό με την ταυτόχρονη αύξηση της ταχύτητας επικοινωνίας μεταξύ των επεξεργαστών. Γενικά, δεν είναι δυνατό να παρέχεται η δυνατότητα άμεσης σύνδεσης κάθε ζεύγους επεξεργαστών, ιδιαίτερα όταν πρόκειται για μεγάλα συστήματα πολυ-υπολογιστών, εφόσον κάτι τέτοιο θα σήμαινε την ύπαρξη n^2 γραμμών επικοινωνίας για n επεξεργαστές. Έτσι, προκειμένου το κόστος να κυμαίνεται σε λογικά επίπεδα και ταυτόχρονα να διασφαλίζεται η δυνατότητα εύκολης αναβάθμισης του συστήματος, μέσω της προσθήκης επιπλέον επεξεργαστών, θα πρέπει το πλήθος των γραμμών επικοινωνίας μεταξύ των επεξεργαστών να είναι είτε σταθερό και ανεξάρτητο από το συνολικό πλήθος των επεξεργαστών, ή στη χειρότερη περίπτωση να προσδιορίζεται ως λογαριθμική συνάρτηση του πλήθους των επεξεργαστών. Οι περισσότερο σημαντικές τοπολογίες συστημάτων πολυ-υπολογιστών είναι : Η *γραμμική* (line), του *δακτυλίου* (ring), η *πλεγματοειδής* (mesh), η *κυλινδρική* (torus) και η *υπερκυβική* (hypercube), οι οποίες θα παρουσιαστούν συνοπτικά στις επόμενες Παραγράφους.

1.4.1 Τοπολογίες Πολυ-υπολογιστών

Στα συστήματα πολυ-υπολογιστών, όπως και σε όλες τις αρχιτεκτονικές υπολογιστών, υπάρχει ένα αντιστάθμισμα (trade-off) μεταξύ κόστους και απόδοσης, όσον αφορά στο σχεδιασμό του δικτύου διασύνδεσης των επεξεργαστών. Έτσι, η καθυστέρηση που υφίσταται ένα μήνυμα που προωθείται μεταξύ των επεξεργαστών εξαρτάται από το πλήθος των γραμμών επικοινωνίας που διανύει. Το πλήθος αυτό καθορίζεται από την εκάστοτε χρησιμοποιούμενη τοπολογία, η οποία αντίστοιχα χαρακτηρίζεται από δύο, κυρίως, παραμέτρους : Τη *συνεκτικότητά* της, η οποία εκφράζει το πλήθος των άμεσων γραμμών επικοινωνίας που διαθέτει κάθε επεξεργαστής και τη *διάμετρο*, η οποία αντιστοιχεί στο μέγιστο πλήθος των ενδιάμεσων γραμμών επικοινωνίας που απαιτείται να διανύσει ένα μήνυμα προκειμένου να επικοινωνήσουν οι δύο πλέον απομακρυσμένοι επεξεργαστές του συστήματος. Οι απλούστερες τοπολογίες έχουν χαμηλή συνεκτικότητα και μεγάλη διάμετρο, ενώ οι περισσότερο σύνθετες τοπολογίες έχουν υψηλή συνεκτικότητα και μικρή διάμετρο. Κάθε τοπολογία έχει τα δικά της χαρακτηριστικά απόδοσης και γι' αυτό το λόγο ο εκάστοτε παράλληλος αλγόριθμος θα πρέπει να προσαρμόζεται ώστε να ταιριάζει στην τοπολογία που χρησιμοποιείται προκειμένου να επιτευχθεί η βέλτιστη δυνατή απόδοση.

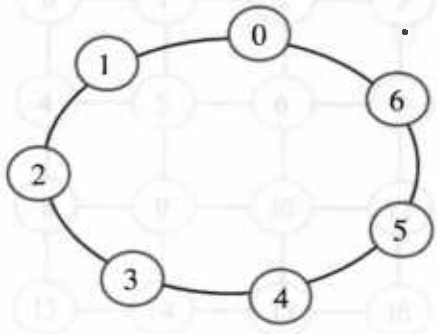
1.4.1.1 Γραμμική Τοπολογία και Τοπολογία Δακτυλίου

Στη *γραμμική* τοπολογία, κάθε επεξεργαστής, εκτός από τον πρώτο και τον τελευταίο, συνδέεται με δύο άμεσα γειτονικούς του επεξεργαστές με τέτοιον τρόπο ώστε όλοι μαζί να σχηματίζουν μία ευθεία γραμμή, όπως φαίνεται στο σχήμα 1.4.1.1-1.

Σχήμα 1.4.1.1-1: Γραμμική Τοπολογία

Μία γραμμική τοπολογία με n επεξεργαστές έχει συνεκτικότητα ίση με 2 και διάμετρο ίση με n-1, ενώ η απόσταση μεταξύ δύο οποιωνδήποτε επεξεργαστών i και j, εκφράζεται από τη σχέση |i-j|.

Η απόδοση της γραμμικής τοπολογίας είναι δυνατό να αυξηθεί σημαντικά, χωρίς καμία περαιτέρω αύξηση του κόστους, αν προστεθεί μία επιπλέον γραμμή επικοινωνίας που να συνδέει τον πρώτο με τον τελευταίο επεξεργαστή. Η τοπολογία που προκύπτει καλείται τοπολογία *δακτυλίου* και απεικονίζεται στο σχήμα 1.4.1.1-2.



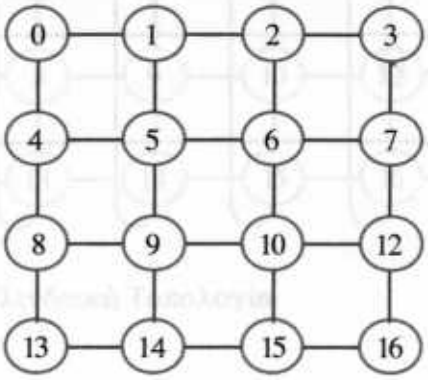
Σχήμα 1.4.1.1-2: Τοπολογία Δακτυλίου

Η επιπλέον επικοινωνιακή σύνδεση που προστίθεται μειώνει τη μέση απόσταση μεταξύ των επεξεργαστών στο μισό, σε σχέση με τη γραμμική τοπολογία, και έτσι μία τοπολογία δακτυλίου με n επεξεργαστές έχει διάμετρο ίση με n/2.

1.4.1.2 Πλεγματοειδής και Κυλινδρική Τοπολογία

Αυξάνοντας το πλήθος των γραμμών επικοινωνίας κάθε επεξεργαστή είναι δυνατό να μειώσουμε τη διάμετρο του δικτύου διασύνδεσης και κατ' επέκταση τη μέση καθυστέρηση επικοινωνίας μεταξύ των επεξεργαστών. Μία τέτοια δυνατότητα παρέχει και η διδιάστατη πλεγματοειδής τοπολογία στην οποία η διάταξη των επεξεργαστών έχει τη μορφή ενός διδιάστατου πίνακα. Σε αυτή την τοπολογία, κάθε επεξεργαστής ο οποίος βρίσκεται στη γραμμή i και στη στήλη j, συνδέεται άμεσα με τέσσερις γειτονικούς του επεξεργαστές, οι οποίοι έχουν συντεταγμένες (i-1,j), (i+1,j), (i,j-1) και (i,j+1). Όλες οι γραμμές επικοινωνίας είναι οριζόντιες μεταξύ γειτονικών στηλών και κατακόρυφες μεταξύ γειτονικών γραμμών του πίνακα των επεξεργαστών, ενώ δεν υπάρχουν καθόλου διαγώνιες γραμμές επικοινωνίας. Επίσης, κάθε επεξεργαστής συνδέεται με τέσσερις γειτονικούς του επεξεργαστές, εκτός από τους συνοριακούς επεξεργαστές οι οποίοι συνδέονται είτε με τρείς ή με δύο μόνον γειτονικούς επεξεργαστές. Οι επεξεργαστές συνήθως αριθμούνται κατά αύξουσα σειρά, από γραμμή σε γραμμή (row major order), ενώ σε μία τέτοια τοπολογία διαστάσεων (mxm) η διάμετρος ισούται με 2(m-1) και αντιστοιχεί στο

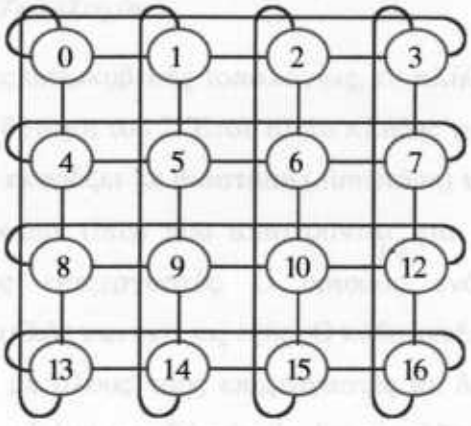
μήκος της διαδρομής που συνδέει τους επεξεργαστές που βρίσκονται στα δύο αντίθετα άκρα του πίνακα. Η διαγραμματική αναπαράσταση μιας διδιάστατης πλεγματοειδούς τοπολογίας απεικονίζεται στο σχήμα 1.4.1.2-1



Σχήμα 1.4.1.2-1: Διδιάστατη Πλεγματοειδής Τοπολογία

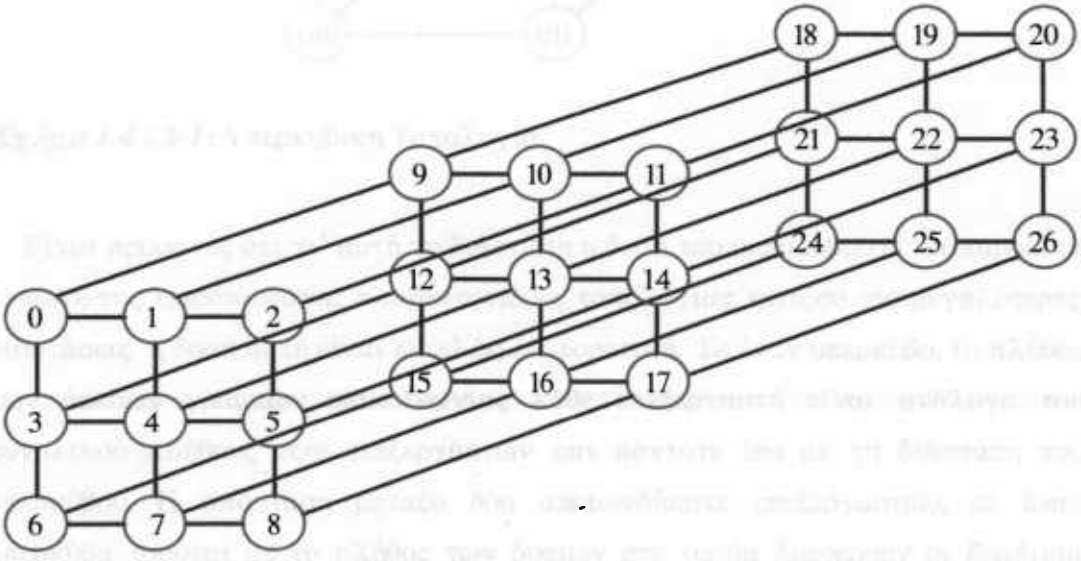
Σε μία πλεγματοειδή τοπολογία, κάθε γραμμή και κάθε στήλη επεξεργαστών έχει τη δομή μιας γραμμικής τοπολογίας. Αν, λοιπόν, σε κάθε τέτοια γραμμική τοπολογία που σχηματίζει κάθε γραμμή και κάθε στήλη του πίνακα, θεωρήσουμε ότι ο πρώτος επεξεργαστής συνδέεται με τον τελευταίο, τότε η τοπολογία που προκύπτει ονομάζεται *κυλινδρική*. Μία τέτοια τοπολογία διαστάσεων (mxm), η οποία περιγράφεται στο σχήμα 1.4.1.2-2, έχει συνεκτικότητα ίση με 4 και διάμετρο ίση με m, η οποία αντιστοιχεί στις επικοινωνιακές διαδρομές που συνδέουν έναν γωνιακό επεξεργαστή με τον αντίστοιχο κεντρικό.

Γενικά, το πλεγματοειδές αυτό μοντέλο είναι δυνατό να επεκταθεί και στις τρεις διαστάσεις, δημιουργώντας έτσι μία τρισδιάστατη πλεγματοειδή τοπολογία. Μία τέτοια τοπολογία μπορεί να θεωρηθεί ως μία ακολουθία διδιάστατων πλεγματοειδών τοπολογιών, οι οποίες καλούνται *επίπεδα* (planes). Τα επίπεδα αυτά είναι διατεταγμένα το ένα πίσω από το άλλο, με τους αντίστοιχους επεξεργαστές τους να συνδέονται μέσω γραμμών επικοινωνίας για κάθε ζεύγος διαδοχικών επιπέδων. Κάθε επεξεργαστής, σε μία τρισδιάστατη πλεγματοειδή τοπολογία, έχει συντεταγμένες (i,j,k), όπου i είναι ο αριθμός της γραμμής, j είναι ο αριθμός της στήλης και k είναι ο αριθμός του επιπέδου στο οποίο βρίσκεται. Κάθε μη συνοριακός επεξεργαστής αυτής της τοπολογίας συνδέεται άμεσα τόσο με τους



Σχήμα 1.4.1.2-2: Κυλινδρική Τοπολογία

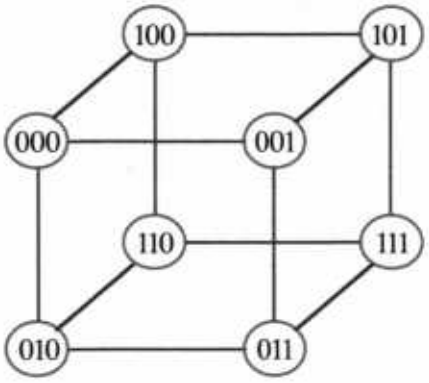
τέσσερις γειτονικούς του επεξεργαστές στο ίδιο επίπεδο, όπως συμβαίνει και στην περίπτωση της διδιάστατης πλεγματοειδούς τοπολογίας, όσο και με τους αντίστοιχούς του επεξεργαστές του προηγούμενου και του επόμενου επιπέδου, δηλαδή ο επεξεργαστής (i,j,k) συνδέεται άμεσα με τους επεξεργαστές (i-1,j,k), (i+1,j,k), (i,j-1,k), (i,j+1,k), (i,j,k-1) και (i,j,k+1). Η διάμετρος μιας τρισδιάστατης πλεγματοειδούς τοπολογίας με m^3 επεξεργαστές ισούται με 3(m-1). Στο σχήμα 1.4.1.2-3 δίνεται η διαγραμματική απεικόνιση μιας τέτοιας τοπολογίας.



Σχήμα 1.4.1.2-3: Τρισδιάστατη Πλεγματοειδής Τοπολογία (m=3)

1.4.1.3 Υπερκυβική Τοπολογία

Στην περίπτωση της υπερκυβικής τοπολογίας, το πλήθος των επεξεργαστών είναι πάντοτε ίσο με μία δύναμη του 2. Έτσι, αν το πλήθος των επεξεργαστών ισούται με 2^d, το d αφενός μεν εκφράζει τη διάσταση (dimension) του υπερκύβου και αφετέρου το πλήθος των δυφίων (bits) που απαιτούνται για τη δυαδική αναπαράσταση καθενός από τους επεξεργαστές. Ο ορισμός ενός υπερκυβικού σχήματος διασύνδεσης είναι απλός και έχει ως εξής : Ο κάθε επεξεργαστής με δυαδικό αριθμό i, συνδέεται άμεσα με όλους τους επεξεργαστές με δυαδικό αριθμό j, όπου το j διαφέρει από το i κατά ένα ακριβώς δυφίο. Για παράδειγμα, ένας υπερκύβος με d=3, θα έχει οκτώ επεξεργαστές, ενώ η δυαδική αναπαράσταση καθενός από αυτούς θα είναι : 000, 001, 010, 011, 100, 101, 110, 111. Με βάση τα παραπάνω ο επεξεργαστής 000 θα συνδέεται άμεσα με τους επεξεργαστές 001, 010 και 100. Το σχήμα 1.4.1.3-1 απεικονίζει μία τέτοια υπερκυβική τοπολογία με διάσταση 3.



Σχήμα 1.4.1.3-1: Υπερκυβική Τοπολογία

Είναι προφανές ότι γι' αυτή τη διάσταση η δομή του υπερκύβου είναι παρόμοια με αυτή της τρισδιάστατης πλεγματοειδούς τοπολογίας, ωστόσο για μεγαλύτερες, διαστάσεις η δομή αυτή είναι εντελώς διαφορετική. Σε έναν υπερκύβο, το πλήθος των άμεσων γραμμών επικοινωνίας κάθε επεξεργαστή είναι ανάλογο του συνολικού πλήθους των επεξεργαστών και πάντοτε ίσο με τη διάσταση του υπερκύβου. Η απόσταση μεταξύ δύο οποιωνδήποτε επεξεργαστών, σε έναν υπερκύβο, ισούται με το πλήθος των δυφίων στα οποία διαφέρουν οι δυαδικοί αριθμοί που αναπαριστούν αυτούς τους επεξεργαστές. Γενικά, σε έναν υπερκύβο με διάσταση d, ισχύει ότι για όλους τους επεξεργαστές υπάρχουν d άμεσα γειτονικοί επεξεργαστές, αντίστοιχα, ενώ η μέγιστη απόσταση μεταξύ οποιουδήποτε ζεύγους

επεξεργαστών έχει, επίσης, μήκος ίσο με d. Με άλλα λόγια, η συνεκτικότητα και η διάμετρος του υπερκύβου ισούνται με τη διάστασή του.

Στον επόμενο πίνακα συνοψίζονται οι τιμές των δύο αυτών παραμέτρων για κάθε τοπολογία που περιγράφηκε. Επίσης, θα πρέπει να σημειωθεί ότι η συνεκτικότητα αποτελεί τον ενδεικτικό παράγοντα έκφρασης του σχετικού κόστους μιας τοπολογίας, ενώ η διάμετρος απότελεί τον ενδεικτικό παράγοντα έκφρασης της σχετικής της απόδοσης.

Τοπολογία	**Συνεκτικότητα**	**Διάμετρος**
Γραμμική	2	n-1
Δακτυλίου	2	n/2
Διδιάστατη Πλεγματοειδής	2-4	$2(n^{1/2}-1)$
Κυλινδρική	4	$n^{1/2}$
Τρισδιάστατη Πλεγματοειδής	3-6	$3(n^{1/3}-1)$
Υπερκυβική	$\log_2 n$	$\log_2 n$

Στον παραπάνω πίνακα οι τοπολογίες παρουσιάζονται κατά αύξουσα τάξη κόστους και απόδοσης.

1.5 Σύγκριση Συστημάτων Πολλαπλών Μονάδων Επεξεργασίας

Καθεμία από τις δύο μεγάλες κατηγορίες παράλληλων συστημάτων που παρουσιάστηκαν, διαθέτει κάποια χαρακτηριστικά τα οποία τις καθιστούν κατάλληλες για την υλοποίηση συγκεκριμένων τύπων αλγορίθμων. Έτσι, αν και γενικά φαίνεται να είναι ευρέως αποδεκτό το γεγονός ότι η χρήση καταμεριζόμενης μνήμης παρέχει την πλέον φυσική λύση όσον αφορά στα προβλήματα επικοινωνίας των παράλληλων αλγορίθμων, ωστόσο υπάρχουν

αρκετοί λόγοι, οι οποίοι καθιστούν το μοντέλο κατανεμημένης μνήμης ως το πλέον υποσχόμενο για να καλύψει τις απαιτήσεις για υψηλού επιπέδου παραλληλισμό (massive parallelism) ορισμένων προβλημάτων. Οι λόγοι αυτοί είναι δυνατό να συνοψιστούν ως εξής:

- Καθώς αυξάνεται το πλήθος των επεξεργαστών, αυξάνεται ταυτόχρονα και το πλήθος των προσπελάσεων στην καταμεριζόμενη μνήμη και κατ' επέκταση το πλήθος των συγκρούσεων, με αποτέλεσμα η καταμεριζόμενη μνήμη να αποτελεί το σημείο συμφόρησης (bottleneck) ολόκληρου του συστήματος.
- Συχνά η πολυπλοκότητα μιας δομής διασύνδεσης, η οποία θα είναι ικανή να χειριστεί τον επικοινωνιακό φόρτο μεταξύ των επεξεργαστών και της μνήμης, είναι εξαιρετικά μεγάλη, γεγονός που αυξάνει σημαντικά το κόστος κατασκευής του συστήματος, ιδιαίτερα όταν το πλήθος των διαθέσιμων επεξεργαστών είναι μεγάλο. Αυτή η δομή διασύνδεσης, στις περισσότερες περιπτώσεις, αποτελεί μία πηγή συγκρούσεων μεταξύ των επεξεργαστών.
- Τέλος, η ύπαρξη προβλημάτων συγχρονισμού, τόσο σε επίπεδο δεδομένων, όσο και σε επίπεδο διεργασιών επιβάλλει περιορισμούς στον προγραμματισμό σε συστήματα πολυεπεξεργασίας, οι οποίοι, συνήθως, υποβαθμίζουν την παράλληλη υλοποίηση και προκαλούν περισσότερα προγραμματιστικά λάθη.

Ένα, επιπλέον, μειονέκτημα των συστημάτων πολυεπεξεργασίας είναι η έλλειψη δυνατότητας αναβάθμισής τους, το οποίο, σε συνδυασμό με τη μικρή, σχετικά, ανοχή καθυστέρησης (latency torelance) σε περίπτωση απομακρυσμένης προσπέλασης της μνήμης, περιορίζει σημαντικά τη δυνατότητα βέλτιστης εκμετάλλευσης του παρεχόμενου επιπέδου παραλληλισμού.

Στην περίπτωση των συστημάτων κατανεμημένης μνήμης εξαλείφονται, βέβαια, οι συγκρούσεις που προκύπτουν από τη συνδρομική προσπάθεια πολλών επεξεργαστών να προσπελάσουν την καταμεριζόμενη μνήμη, αλλά εξακολουθεί να υφίσταται το πρόβλημα της ύπαρξης πιθανών συγκρούσεων σε επίπεδο επικοινωνιακού δικτύου διασύνδεσης. Το πρόβλημα αυτό διογκώνεται καθώς αυξάνεται το πλήθος των άμεσων συνδέσεων μεταξύ των επεξεργαστών. Ωστόσο, αν η τοπολογία των επεξεργαστών αντανακλά τη δομή των αλγοριθμικών υπολογισμών, τότε ο φόρτος της δρομολόγησης είναι δυνατόν να διατηρηθεί σε λογικά επίπεδα. Η τελευταία αυτή παρατήρηση τεκμηριώνει την άποψη ότι η αρχιτεκτονική καταμεριζόμενης μνήμης είναι περισσότερο κατάλληλη για την

υλοποίηση γενικής μορφής υπολογισμών, ενώ η αρχιτεκτονική κατανεμημένης μνήμης, ανάλογα με την τοπολογία διασύνδεσης των επεξεργαστών, μπορεί να είναι περισσότερο αποδοτική όσον αφορά στην υλοποίηση συγκεκριμένου τύπου υπολογισμών.

Όσον αφορά στην απόδοση των προγραμμάτων τα οποία αναπτύσσονται σε συστήματα πολυ-υπολογιστών, αυτή εξαρτάται άμεσα από την τοποθέτηση των δεδομένων στις διάφορες τοπικές μνήμες, αφού η διαδικασία προσπέλασης μιας τοπικής μνήμης μπορεί να είναι σημαντικά ταχύτερη από τη χρονική καθυστέρηση που εισάγει το επικοινωνιακό δίκτυο. Ωστόσο, η ανάπτυξη αποδοτικών προγραμμάτων σε τέτοιου είδους συστήματα προυποθέτει και τη λεπτομερειακή γνώση της αρχιτεκτονικής της μηχανής, ώστε να είναι δυνατόν να εκφραστούν οι αλγόριθμοι με τρόπο που να εκμεταλλεύονται πλήρως τις ιδιαιτερότητες της εκάστοτε μηχανής. Συχνά, τα προγράμματα τα οποία παρουσιάζουν υψηλή απόδοση σε συστήματα πολυεπεξεργασίας υστερούν σε απόδοση όταν υλοποιούνται σε συστήματα πολυ-υπολογιστών, εξαιτίας της διαφορετικής αρχιτεκτονικής οργάνωσης των τελευταίων, η οποία εισάγει ένα διαφορετικό σύνολο παραγόντων που επηρεάζουν τη συνολική απόδοση των παράλληλων προγραμμάτων.

Απαραίτητη προυπόθεση, όμως, για τη βέλτιστη δυνατή εκμετάλλευση των παρεχόμενων δυνατοτήτων παραλληλισμού από οποιοδήποτε σύστημα, είναι ο εντοπισμός των ιδιαίτερων χαρακτηριστικών του εκάστοτε παράλληλου αλγόριθμου, τα οποία μπορεί να συμβάλλουν στη μείωση της συνολικής απόδοσης και η υιοθέτηση με βάση αυτά, της καλύτερης δυνατής αρχιτεκτονικής για την υλοποίηση του αλγόριθμου αυτού. Η εφαρμογή της διαδικασίας αυτής κάνοντας χρήση ενός ικανού συνόλου αλγορίθμων, επιτρέπει, αφενός μεν, την αποτίμηση των συστημάτων και, αφετέρου, μία κατάλληλη αντιστοίχιση μεταξύ αλγορίθμων και αρχιτεκτονικών συστημάτων πολλαπλών επεξεργαστών, προκειμένου για την επίτευξη της μέγιστης δυνατής απόδοσης.

1.6 Παράλληλος Προγραμματισμός

Με την εμφάνιση των παράλληλων συστημάτων πολλαπλών επεξεργαστών προέκυψαν νέου είδους απαιτήσεις που αφορούσαν στην ανάπτυξη προγραμμάτων σε αυτά τα συστήματα. Σε αντίθεση με ένα σειριακό πρόγραμμα, το οποίο

αποτελείται από μία ακολουθία εντολών τις οποίες πρέπει να εκτελέσει ο σειριακός υπολογιστής, ένα παράλληλο πρόγραμμα πρέπει να περιέχει μία τέτοια ακολουθία εντολών για κάθε ξεχωριστό επεξεργαστή, καθώς επίσης και εντολές για το συντονισμό και την αλληλεπίδραση των επεξεργαστών. Αυτή ακριβώς η αναγκαιότητα της δημιουργίας και του συντονισμού πολλών παράλληλων υπολογιστικών διεργασιών προσθέτει μία νέα διάσταση στη διαδικασία ανάπτυξης προγραμμάτων. Οι σειριακοί αλγόριθμοι, που αφορούν σε συγκεκριμένα προβλήματα, πρέπει πλέον να ανασκευαστούν, έτσι ώστε να δημιουργηθούν πολλά παράλληλα ρεύματα εντολών τα οποία θα εκτελούνται σε διαφορετικούς επεξεργαστές. Κατά συνέπεια, μολονότι οι αρχιτεκτονικές πολυεπεξεργασίας και πολυ-υπολογιστών παρέχουν τη δυνατότητα σημαντικής αύξησης της υπολογιστικής ισχύος με μικρό σχετικά κόστος κατασκευής, ωστόσο η πραγματική αξιοποίηση αυτής της δυνατότητας καθίσταται εφικτή μόνο μέσω της κατανόησης των παράλληλων γλωσσών προγραμματισμού και του σχεδιασμού παράλληλων αλγορίθμων.

Η έννοια της *διεργασίας* (process) είναι θεμελιώδης για την ανάπτυξη παράλληλων προγραμμάτων και αναφέρεται στην ακολουθία εκείνη των εντολών οι οποίες μπορούν να εκτελεστούν από ένα μεμονωμένο επεξεργαστή. Έτσι, η διεργασία θεωρείται ως η βασική δομική μονάδα των παράλληλων προγραμμάτων, η οποία συχνά ταυτίζεται με την έννοια μιας *υπορουτίνας* (subroutine) ή μιας *διαδικασίας* (procedure) που εκτελείται σε ένα συγκεκριμένο φυσικό επεξεργαστή. Η διαθεσιμότητα μεγάλου πλήθους φυσικών επεξεργαστών, σε ένα σύστημα, συνεπάγεται τη δυνατότητα παράλληλης εκτέλεσης ενός μεγάλου πλήθους διεργασιών, καθεμία από τις οποίες υλοποιεί ένα τμήμα του συνολικού υπολογιστικού έργου του εκάστοτε αλγόριθμου, γεγονός που μειώνει σημαντικά τη συνολική χρονική πολυπλοκότητα του τελευταίου. Γι' αυτό το λόγο, η έννοια της διεργασίας αποτελεί βασικό χαρακτηριστικό όλων των παράλληλων γλωσσών προγραμματισμού και αποσκοπεί στο να παρέχει τη δυνατότητα στον εκάστοτε προγραμματιστή παράλληλων αλγορίθμων να διαμορφώνει τον κώδικα των προγραμμάτων του, με τέτοιον τρόπο ώστε να αξιοποιεί την υπολογιστική ισχύ των διαθέσιμων επεξεργαστών του συστήματος που χρησιμοποιεί.

Η ενσωμάτωση της έννοιας της διεργασίας σε μία παράλληλη γλώσσα προγραμματισμού προυποθέτει την ύπαρξη κάποιου μηχανισμού, ο οποίος θα

χρησιμοποιείται για τον ορισμό και τη δημιουργία των διεργασιών. Επίσης, η γλώσσα προγραμματισμού θα πρέπει να διαθέτει και κάποια χαρακτηριστικά τα οποία θα επιτρέπουν την πρόσβαση των παράλληλων επεξεργαστών σε κοινές δομές δεδομένων, έτσι ώστε να είναι εφικτή η αλληλεπίδραση των διαφόρων διεργασιών κατά τη διάρκεια που αυτές εκτελούν επιμέρους κομάτια του συνολικού προγράμματος. Ένα ακόμη βασικό χαρακτηριστικό των παράλληλων γλωσσών προγραμματισμού αφορά στη δυνατότητά τους να παρέχουν συγχρονισμό των διεργασιών, εφόσον αυτές εξαιτίας του ότι εκτελούνται σε διαφορετικούς φυσικούς επεξεργαστές διαφοροποιούνται ως προς τον απαιτούμενο χρόνο υλοποίησής τους. Οι έννοιες των παράλληλων διεργασιών και του ελέγχου της αλληλεπίδρασης των διεργασιών πρωτοεμφανίστηκαν στα τέλη της δεκαετίας του 1960, όταν αναπτύχθηκαν τα πρώτα συστήματα *πολυπρογραμματισμού* (multiprogramming) και *χρονικού καταμερισμού* (time sharing). Τα συστήματα αυτά επέτρεπαν σε πολλά προγράμματα να είναι ενεργά κατά τη διάρκεια της ίδιας χρονικής περιόδου, εφαρμόζοντας χρονικό καταμερισμό, και έτσι η έννοια της παράλληλης διεργασίας αποτέλεσε μία πολύ χρήσιμη οργανωτική αρχή για το σχεδιασμό των λειτουργικών συστημάτων αυτών των υπολογιστών, παρόλο που το υλικό τους συνήθως αποτελείτο από έναν μόνο φυσικό επεξεργαστή.

Κατά τη διάρκεια της δεκαετίας του 1970 αναπτύχθηκαν αρκετές γλώσσες προγραμματισμού, οι οποίες περιελάμβαναν χαρακτηριστικά δημιουργίας και αλληλεπίδρασης διεργασιών και οι οποίες είχαν εφαρμογή στην ανάπτυξη λειτουργικών συστημάτων. Μερικές από αυτές τις γλώσσες προγραμματισμού που αναπτύχθηκαν είναι, η Concurrent Pascal (Hansen, [42]), η Modula (Wirth, [70]), η PLITS (Feldman, [36]) κ.ά. Η έννοια της διεργασίας βρήκε επίσης εφαρμογή και στα ενσωματωμένα συστήματα πραγματικού χρόνου και γι' αυτό συμπεριλήφθηκε στη γλώσσα ADA. Τέλος, η γλώσσα Argus (Liskov, et al, [54]) χρησιμοποιούσε παράλληλες διεργασίες για την ανάπτυξη συστημάτων κατανεμημένων βάσεων δεδομένων για δίκτυα υπολογιστών.

Καθώς τα συστήματα πολυεπεξεργασίας και πολυ-υπολογιστών γίνονταν ευρέως γνωστά, κατά τη διάρκεια της δεκαετίας του 1980, η έννοια της διεργασίας προστέθηκε και σε πολλές γλώσσες προγραμματισμού οι οποίες χρησιμοποιούνταν για την ανάπτυξη εφαρμογών, όπως, για παράδειγμα, η Fortran, η C, η Pascal και η Lisp. Στα περισσότερα εμπορικά συστήματα αυτό πραγματοποιήθηκε μέσω της

χρήσης ειδικών κλήσεων του συστήματος (special system calls) από τη γλώσσα προγραμματισμού, προκειμένου για τη δημιουργία, την επικοινωνία και το συγχρονισμό των διεργασιών. Λόγω της απουσίας τυπικών παράλληλων γλωσσών προγραμματισμού, κάθε κατασκευαστής συστημάτων υπολογιστών, δημιουργούσε τη δική του μοναδική παραλλαγή τέτοιου είδους κλήσεων λειτουργικού συστήματος. Από την άλλη μεριά, αρκετές προτάσεις έγιναν για την ανάπτυξη τυπικών παράλληλων γλωσσών προγραμματισμού, ενώ δύο από αυτές, η Linda και η Multilisp, υλοποιήθηκαν σε διάφορα εμπορικά συστήματα. Θα πρέπει να σημειωθεί, ότι παρόλο που υπάρχουν σημαντικές διαφορές, όσον αφορά στην ακριβή σύνταξη και σημασιολογία των χαρακτηριστικών των παράλληλων γλωσσών προγραμματισμού που χρησιμοποιούνται σήμερα από τα διάφορα συστήματα πολυεπεξεργασίας και πολυ-υπολογιστών, στην πραγματικότητα όλα αυτά τα χαρακτηριστικά βασίζονται σε μερικές στοιχειώδεις και απλές ιδέες, στις οποίες η έννοια της διεργασίας αποτελεί το βασικό πυρήνα.

Είναι προφανές, βέβαια, ότι καμία ταύτιση δεν υπάρχει μεταξύ των εννοιών διεργασία και επεξεργαστής. Ο επεξεργαστής είναι μία φυσική συσκευή υλικού, η οποία έχει την ικανότητα να προσπελάζει τη μνήμη και να υπολογίζει νέες τιμές δεδομένων, ενώ η διεργασία είναι μία περισσότερο αφηρημένη έννοια η οποία περιγράφει κάποια υπολογιστική δραστηριότητα. Έτσι, προκειμένου ένας φυσικός επεξεργαστής να υλοποιήσει έναν υπολογισμό θα πρέπει να του *'εκχωρηθεί'* κάποια διεργασία, δηλαδή θα πρέπει να του ανατεθεί να υλοποιήσει κάποιο υπολογιστικό έργο. Η διεργασία που ανατίθεται σε κάθε επεξεργαστή καθορίζεται από το πρόγραμμα που αναπτύσσεται, γεγονός που καθιστά προφανές ότι ένας επεξεργαστής αφορά στο υλικό ενός Η/Υ, ενώ μία διεργασία αφορά στο λογισμικό του.

1.7 Παράλληλοι Αλγόριθμοι και Τεχνικές Παραλληλισμού

Η πραγματική αξιοποίηση της τεράστιας υπολογιστικής ισχύος που προσφέρεται εξαιτίας της ανάπτυξης της τεχνολογίας παράλληλης επεξεργασίας, επιτυγχάνεται μόνον μέσω του σχεδιασμού παράλληλων αλγορίθμων, οι οποίοι χρησιμοποιούν έναν μεγάλο αριθμό επεξεργαστών που εργάζονται παράλληλα για την ολοκλήρωση ενός ενιαίου υπολογιστικού έργου. Σε ορισμένες περιπτώσεις, οι

τυπικοί σειριακοί αλγόριθμοι είναι δυνατό να προσαρμοστούν εύκολα, έτσι ώστε να μπορούν να υλοποιηθούν σε παράλληλα συστήματα. Τις περισσότερες φορές, όμως, απαιτείται η εξαρχής ανάλυση του ίδιου του προβλήματος προκειμένου να οδηγηθεί κανείς στο σχεδιασμό των κατάλληλων παράλληλων αλγορίθμων. Κατά τη διάρκεια της τελευταίας 20-ετίας, ο τομέας του σχεδιασμού παράλληλων αλγορίθμων έχει επικεντρώσει το ενδιαφέρον πολλών επιστημόνων, οι οποίοι έχουν προτείνει παράλληλους αλγόριθμους για ένα ευρύ φάσμα πρακτικών προβλημάτων, τα οποία περιλαμβάνουν την ταξινόμηση, την επεξεργασία γραφημάτων, την επίλυση συστημάτων γραμμικών εξισώσεων, την επίλυση διαφορικών εξισώσεων και την προσομοίωση. Τελευταία, οι επιστημονικές προσπάθειες έχουν επικεντρωθεί στην ανάπτυξη αποδοτικών παράλληλων προγραμμάτων ειδικά για αρχιτεκτονικές πολυεπεξεργασίας και πολυ-υπολογιστών.

Από μία περισσότερο προσεκτική εξέταση της ποικιλίας των παράλληλων αλγορίθμων οι οποίοι έχουν αναπτυχθεί μέχρι σήμερα, διαφαίνεται η χρήση συγκεκριμένων μοντέλων, τα οποία βασίζονται σε μερικές απλές και γενικές οργανωτικές τεχνικές, οι οποίες δεν προσανατολίζονται σε συγκεκριμένα πεδία εφαρμογής, αλλά αποτελούν πρακτικές προγραμματιστικές τεχνικές που μπορούν να εφαρμοστούν σε πολλές και ποικίλες εφαρμογές. Στη συνέχεια, παρουσιάζεται μία κατηγοριοποίηση των παράλληλων προγραμματιστικών τεχνικών, μαζί με μία συνοπτική επεξήγηση καθεμιάς από αυτές.

1.7.1 Παραλληλισμός Δεδομένων (Data Parallelism)

Η περισσότερο ευρέως χρησιμοποιούμενη τεχνική για την ανάπτυξη παράλληλων προγραμμάτων είναι ο *παραλληλισμός δεδομένων*, ο οποίος γενικά αφορά, στην παράλληλη εφαρμογή της ίδιας λειτουργίας σε κάθε στοιχείο μιας δομής δεδομένων. Αυτός ο τύπος παραλληλισμού είναι ιδιαίτερα χρήσιμος προκειμένου για αριθμητικούς αλγόριθμους οι οποίοι επεξεργάζονται τα στοιχεία μεγάλου μεγέθους πινάκων και διανυσμάτων. Ο πυρήνας των πιο παραδοσιακών σειριακών προγραμμάτων, που υλοποιούν τέτοιου είδους αλγόριθμους, αποτελείται από μία σειρά από εμφωλιασμένους βρόχους (nested loops), οι οποίοι εκτελούν σύνθετες εντολές πάνω στα δεδομένα αυτών των πινάκων, προκειμένου να ληφθεί κάποιο αριθμητικό αποτέλεσμα. Σήμερα, τα πιο πρακτικά παράλληλα προγράμματα

δημιουργούνται μέσω της αναπροσαρμογής αυτών των σειριακών αλγορίθμων με τέτοιο τρόπο, ώστε να επιτυγχάνεται ο παραλληλισμός των εμφωλιασμένων βρόχων που περιέχουν. Κάτι τέτοιο συνεπάγεται ότι ο παραλληλισμός σε αυτού του είδους τα προγράμματα προκύπτει από την παράλληλη εφαρμογή της ίδιας λειτουργίας σε διαφορετικά τμήματα των πινάκων δεδομένων.

Η έννοια του παραλληλισμού των δεδομένων είναι απλή και υπονοεί την αντιστοίχιση της δομής του παραλληλισμού σε αυτή των δεδομένων. Σε ορισμένες περιπτώσεις ο παραλληλισμός εφαρμόζεται σε επίπεδο μεμονωμένων στοιχείων, ενώ σε άλλες οι εκάστοτε χρησιμοποιούμενες δομές δεδομένων υποδιαιρούνται σε κατάλληλου μεγέθους ομάδες δεδομένων, οι οποίες στη συνέχεια υφίστανται παράλληλη επεξεργασία. Η εξοικείωση των προγραμματιστών σειριακών προγραμμάτων με τη χρήση εμφωλιασμένων βρόχων, αποτέλεσε το βασικότερο παράγοντα της επικράτησης αυτής της μορφής παραλληλισμού στα πρώτα στάδια ανάπτυξης της παράλληλης επεξεργασίας. Ωστόσο, όπως αποδεικνύεται μέσα από πολλά παραδείγματα, πολλοί επαναληπτικοί βρόχοι δεν επιδέχονται άμεσα την παράλληλη προσαρμογή τους, εξαιτίας του ότι συχνά απαιτείται η συσσώρευση κάποιων προσωρινών τιμών μεταξύ των επαναλήψεων, γεγονός που επιβάλλει τη σειριακή εκτέλεσή τους. Σε αυτές τις περιπτώσεις, η σειριακή έκδοση του αλγόριθμου θα πρέπει να τροποποιηθεί ριζικά προκειμένου να προκύψει μία κατάλληλη παράλληλη έκδοση, γεγονός που συχνά συνεπάγεται τον εκ νέου σχεδιασμό ολόκληρου του αλγόριθμου. Επίσης, ο παραλληλισμός των επαναληπτικών βρόχων σε συστήματα πολυεπεξεργασίας, συχνά προκαλεί συγκεκριμένα προβλήματα που επηρεάζουν την απόδοση των προγραμμάτων και σχετίζονται με το κόστος δημιουργίας διεργασιών, τον ανταγωνισμό των επεξεργαστών για την προσπέλαση της καταμεριζόμενης μνήμης και τις καθυστερήσεις που προκαλεί η απαίτηση για συγχρονισμό των διεργασιών.

Τέλος, θα πρέπει να σημειωθεί ότι ο παραλληλισμός των δεδομένων αποτελεί περισσότερο μία ευρεία κατηγορία παράλληλων τεχνικών, παρά μία μεμονωμένη προγραμματιστική τεχνική ανάπτυξης παράλληλων προγραμμάτων.

1.7.2 Κατάτμηση των Δεδομένων (Data Partitioning)

Η τεχνική αυτή αποτελεί ουσιαστικά μία υποκατηγορία της τεχνικής του παραλληλισμού δεδομένων, η οποία αφορά στη φυσική κατάτμηση του κοινού

χώρου των δεδομένων σε διακεκριμένα τμήματα, καθένα από τα οποία υφίσταται παράλληλη επεξεργασία από έναν διαφορετικό επεξεργαστή. Η τεχνική αυτή είναι ιδιαίτερα κατάλληλη για συστήματα πολυ-υπολογιστών, στα οποία τα διαφορετικά τμήματα δεδομένων κατανέμονται σε ξεχωριστές τοπικές μνήμες και η υπολογιστική δραστηριότητα κάθε επεξεργαστή αφορά κυρίως στα τοπικά του δεδομένα. Σε αυτή την περίπτωση, μία παράλληλη διεργασία ανατίθεται σε κάθε επεξεργαστή, η οποία 'φορτώνεται' επίσης στην τοπική μνήμη και έτσι του παρέχει τη δυνατότητα άμεσης πρόσβασης ενός συγκεκριμένου τμήματος δεδομένων.

Προκειμένου να επιτευχθεί η βέλτιστη δυνατή απόδοση, κάθε διεργασία θα πρέπει να χρησιμοποιεί, κυρίως, τις δικές της τοπικές μεταβλητές και το δικό της τοπικό τμήμα δεδομένων. Στο βαθμό που μία διεργασία απαιτείται να προσπελάσει δεδομένα τα οποία είναι αποθηκευμένα σε απομακρυσμένες μνήμες, τότε χρησιμοποιείται το δίκτυο διασύνδεσης των επεξεργαστών. Συνήθως η φύση των προβλημάτων είναι τέτοια που καθιστά αδύνατο τον περιορισμό της εμβέλειας μιας διεργασίας αποκλειστικά και μόνο στην προσπέλαση του τοπικού τμήματος δεδομένων. Παρόλα αυτά, ένας παράλληλος αλγόριθμος είναι δυνατόν να οργανωθεί με τέτοιον τρόπο, ώστε οι περισσότεροι υπολογισμοί κάθε διεργασίας να αφορούν στα τοπικά της δεδομένα. Αυτός, άλλωστε, είναι και ο βασικός στόχος των παράλληλων προγραμμάτων που αναπτύσσονται για συστήματα πολυ-υπολογιστών, εφόσον οι προσπάθειες προσπέλασης δεδομένων που βρίσκονται σε απομακρυσμένες μνήμες, εισάγουν σημαντικές καθυστερήσεις που οφείλονται στη χρήση του επικοινωνιακού δικτύου διασύνδεσης.

1.7.3 Χαλαροί Αλγόριθμοι (Relaxed Algorithms)

Η ανάπτυξη τέτοιου είδους αλγορίθμων συνεπάγεται την αυτόνομη εκτέλεση των παράλληλων διεργασιών, με την έννοια ότι δεν υφίστανται προβλήματα συγχρονισμού και επικοινωνίας μεταξύ τους. Οι χαλαροί αλγόριθμοι αποδίδουν εξίσου ικανοποιητικά τόσο σε συστήματα πολυεπεξεργασίας, όσο και σε συστήματα πολυ-υπολογιστών επιτυγχάνοντας αρκετά υψηλή επιτάχυνση. Αυτό οφείλεται στο ότι, μολονότι οι επεξεργαστές θα πρέπει να προσπελάσουν κάποια κοινά δεδομένα, καθένας από αυτούς μπορεί να πραγματοποιεί υπολογισμούς εντελώς αυτόνομα, χωρίς να απαιτεί τη χρήση κάποιων ενδιάμεσων αποτελεσμάτων που παράγονται από άλλους επεξεργαστές. Η πλήρης αυτονομία κάθε επεξεργαστή

αποτελεί και το βασικό πλεονέκτημα αυτής της κατηγορίας αλγορίθμων, το οποίο συμβάλλει αποφασιστικά στον εύκολο προγραμματισμό τους.

1.7.4 Συγχρονισμένες Επαναλήψεις (Synchronous Iteration)

Η πλέον πρακτική εφαρμογή των παράλληλων υπολογισμών αφορά, κυρίως, σε πολύ μεγάλης κλίμακας αριθμητικά προβλήματα, τα οποία επεξεργάζονται, πολυδιάστατους πίνακες κάνοντας χρήση επαναληπτικών μεθόδων. Σε αυτές τις περιπτώσεις σε κάθε νέα επανάληψη, συνήθως, χρησιμοποιούνται τα αποτελέσματα που παράγονται από την προηγούμενη επανάληψη, ενώ το πρόγραμμα συγκλίνει βαθμιαία σε μία λύση που ικανοποιεί την επιθυμητή ακρίβεια. Σε τέτοιου είδους επαναληπτικούς αλγόριθμους υπάρχει η δυνατότητα εφαρμογής της έννοιας του παραλληλισμού κατά τη διάρκεια κάθε επανάληψης, μέσω της δημιουργίας πολλών παράλληλων διεργασιών οι οποίες να επεξεργάζονται διαφορετικά τμήματα δεδομένων. Εξαιτίας της φύσης των προβλημάτων, όμως, οι διεργασίες αυτές θα πρέπει να συγχρονίζονται στο τέλος κάθε επανάληψης, αφού τα αποτελέσματα που παράγονται από μία διεργασία μπορεί να χρησιμοποιούνται από κάποιες άλλες διεργασίες στην επόμενη επανάληψη.

Ο συγχρονισμός των διεργασιών είναι δυνατόν να μειώσει σημαντικά την απόδοση ενός παράλληλου προγράμματος, εφόσον από τη φύση του αντιτίθεται στην έννοια του παραλληλισμού εξαιτίας του ότι επιβάλλει την εφαρμογή κάποιας μορφής κεντρικού ελέγχου σε όλες τις διεργασίες. Το γεγονός αυτό μπορεί να οδηγήσει, περαιτέρω, στην πιθανότητα ύπαρξης ανταγωνισμού και καθυστερήσεων κατά την εκτέλεση των διεργασιών. Για το λόγο αυτό, πολλές διαφορετικές τεχνικές συγχρονισμού έχουν αναπτυχθεί, καθεμία από τις οποίες διαθέτει τα δικά της χαρακτηριστικά απόδοσης. Μερικές από αυτές εισάγουν μία γραμμική καθυστέρηση, ανάλογη του πλήθους των επεξεργαστών, ενώ άλλες, όπως για παράδειγμα, η τεχνική του δυαδικού δένδρου, προκαλεί μία καθυστέρηση η οποία εκφράζεται ως λογαριθμική συνάρτηση του πλήθους των επεξεργαστών. Τέλος, σε συγκεκριμένους αλγόριθμους είναι δυνατή η εφαρμογή τεχνικών τοπικού συγχρονισμού, σύμφωνα με τις οποίες κάθε διεργασία συγχρονίζεται μόνο με μερικές άμεσα γειτονικές της διεργασίες. Σε αυτή την περίπτωση, εισάγεται μία σταθερή καθυστέρηση η οποία είναι ανεξάρτητη του πλήθους των επεξεργαστών.

Θα πρέπει να σημειωθεί ότι οι παράλληλοι αλγόριθμοι, οι οποίοι απαιτούν τη χρήση τεχνικών συγχρονισμού, παρουσιάζουν μεγαλύτερη απόδοση όταν υλοποιούνται σε συστήματα πολυεπεξεργασίας, γιατί στα συστήματα αυτά ο χρόνος που απαιτείται για το συγχρονισμό όλων των επεξεργαστών είναι σχετικά μικρός. Αντίθετα, στα συστήματα πολυ-υπολογιστών, ο χρόνος αυτός είναι πολύ μεγαλύτερος εξαιτίας της κατανεμημένης φύσης των επεξεργαστών. Κατά συνέπεια, η ανάπτυξη τέτοιου είδους προγραμμάτων σε συστήματα πολυ-υπολογιστών θα πρέπει να γίνεται πολύ προσεκτικά στο βαθμό που είναι επιθυμητή η επίτευξη καλής απόδοσης.

1.7.5 Αντίγραφα Εργατών (Replicated Workers)

Στους περισσότερους παράλληλους αλγόριθμους, το πλήθος των υπολογιστικών έργων προς υλοποίηση είναι, συνήθως, γνωστό εκ των προτέρων, με αποτέλεσμα να είναι δυνατή η κατανομή τους στους διάφορους επεξεργαστές του συστήματος. Ωστόσο, υπάρχει και μία σημαντική κατηγορία αλγορίθμων, στους οποίους τα συγκεκριμένα υπολογιστικά έργα αυξομειώνονται δυναμικά, καθώς ο εκάστοτε αλγόριθμος εξελίσσεται, γεγονός που καθιστά αδύνατη την εξαρχής κατανομή τους μεταξύ των επεξεργαστών. Σε αυτή την περίπτωση, προκειμένου να επιτευχθεί μία εξισορρόπηση της κατανομής του φόρτου εργασίας, τα υπολογιστικά έργα θα πρέπει να ανατίθενται στους επεξεργαστές με δυναμικό τρόπο, καθώς παράγονται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Κάτι τέτοιο επιτυγχάνεται κάνοντας χρήση της παράλληλης τεχνικής που καλείται *αντίγραφα εργατών*.

Η συγκεκριμένη τεχνική αφορά στη διατήρηση μιας κεντρικής πισίνας εργασίας (work pool), η οποία διαθέτει μία συλλογή από παρόμοια υπολογιστικά έργα, τα οποία μπορούν να εκτελεστούν από οποιαδήποτε από τις διεργασίες-εργάτες. Όταν μία τέτοια διεργασία καταστεί αδρανής τότε μπορεί να ανακτήσει ένα καινούργιο υπολογιστικό έργο από την πισίνα εργασίας και να εκτελέσει τους κατάλληλους υπολογισμούς. Κατά τη διάρκεια εκτέλεσης αυτών των υπολογισμών, η διεργασία είναι δυνατό να παράγει νέα υπολογιστικά έργα τα οποία προσθέτει στην πισίνα εργασίας. Το συνολικό υπολογιστικό έργο τερματίζεται όταν η πισίνα εργασίας αδειάσει πλήρως.

Τρία βασικά θέματα, τα οποία αφορούν στην απόδοση και τα οποία προκύπτουν από την υλοποίηση αυτής της τεχνικής παραλληλισμού, είναι : Ο ανταγωνισμός, η

εξισορρόπηση της κατανομής του φόρτου εργασίας και ο τερματισμός. Το πρόβλημα του ανταγωνισμού προκύπτει, συνήθως, εξαιτίας της επικέντρωσης του ελέγχου που επιβάλλει αυτή η υλοποίηση, εφόσον η καθολική συλλογή και η διανομή των διαφόρων υπολογιστικών έργων από την πισίνα εργασίας, προυποθέτει την προσπέλασή της από όλες τις διεργασίες-εργάτες. Μία προφανής λύση σε αυτό το πρόβλημα αφορά στη μερική αποκέντρωση της πισίνας εργασίας μέσω της διαίρεσής της σε τμήματα, τα οποία θα μπορούν να προσπελαστούν παράλληλα. Ωστόσο, αυτή η λύση είναι δυνατό να οδηγήσει στη μη εξισορροπημένη κατανομή των υπολογιστικών έργων μεταξύ των διαφόρων τμημάτων της πισίνας εργασίας και κατ' επέκταση στη διαθεσιμότητα μη ισορροπημένου φόρτου εργασίας μεταξύ των επεξεργαστών. Τέλος, ο τερματισμός αποτελεί εξίσου ένα σημαντικό πρόβλημα γιατί εξαρτάται από την κατάσταση της πισίνας εργασίας και των διεργασιών σε μία δεδομένη χρονική στιγμή. Ένας αλγόριθμος που χρησιμοποιεί τη συγκεκριμένη τεχνική παραλληλισμού, θα πρέπει φυσικά να τερματίζει όταν δεν υπάρχει περαιτέρω υπολογιστικό έργο προς υλοποίηση, δηλαδή όταν η πισίνα εργασίας είναι άδεια. Μολονότι, όμως μπορεί η πισίνα εργασίας να είναι άδεια, υπάρχει η περίπτωση κάποιες ενεργές διεργασίες να προτίθενται να προσθέσουν νέα υπολογιστικά έργα σε αυτή. Κατά συνέπεια, η συνθήκη τερματισμού θα πρέπει να προυποθέτει και το άδειασμα της πισίνας εργασίας, αλλά και τη μη περαιτέρω υπολογιστική δραστηριότητα από μέρους των διεργασιών. Η συγκεκριμένη συνθήκη τερματισμού είναι συχνά δύσκολο να ελεγχθεί κατά τη διάρκεια της εκτέλεσης του αλγόριθμου και ιδιαίτερα όταν αυτός υλοποιείται σε ένα σύστημα πολυ-υπολογιστών.

Αν και τα προβλήματα που παρουσιάζει αυτή η τεχνική παραλληλισμού είναι αρκετά σημαντικά, ωστόσο σε πολλές περιπτώσεις η εφαρμογή της κρίνεται αναγκαία και απαραίτητη, ιδιαίτερα όταν πρόκειται για συνδυαστικά προβλήματα, όπως είναι τα προβλήματα ανίχνευσης σε δένδρα και γραφήματα.

1.7.6 Σωληνικοί Υπολογισμοί (Pipelined Computations)

Σε αυτή την περίπτωση, οι διεργασίες διατάσσονται προσομοιώνοντας μία συγκεκριμένη τοπολογία, όπως ειναι η τοπολογία δακτυλίου ή η πλεγματοειδής τοπολογία. Κατόπιν, τα δεδομένα ρέουν διαμέσω της δομής αυτής των διεργασιών, ενώ κάθε διεργασία εκτελεί μία συγκεκριμένη φάση του συνολικού υπολογιστικού

έργου. Οι αλγόριθμοι που αναπτύσσονται εφαρμόζοντας τη συγκεκριμένη τεχνική παραλληλισμού είναι αποδοτικοί κυρίως σε συστήματα πολυ-υπολογιστών, εξαιτίας του κανονικοποιημένου μοντέλου της ροής των δεδομένων και της έλλειψης της απαίτησης για καθολική προσπέλαση των κοινών δεδομένων.

1.8 Ενδεικτικές Παράμετροι Αποτίμησης Απόδοσης

Η παράλληλη επεξεργασία έχει σήμερα καθιερωθεί ως η συνηθισμένη προσέγγιση για την επίτευξη υψηλής απόδοσης. Διάφοροι τύποι παράλληλων συστημάτων υπολογιστών έχουν κατασκευαστεί και πολλοί παράλληλοι αλγόριθμοι έχουν αντίστοιχα αναπτυχθεί για να εκμεταλλευτούν τις δυνατότητες αυτών των μηχανών. Δεν υπάρχουν, όμως, αποτελεσματικές τεχνικές για την αποτίμηση της απόδοσης των παράλληλων αυτών μηχανών και των αντίστοιχων αλγορίθμων. Δηλαδή, δεν υπάρχει καθιερωμένη *μετρική* (metric) παράμετρος αποτίμησης του κέρδους σε απόδοση, που επιτυγχάνεται διαμέσω της παράλληλης επεξεργασίας. Η πιο συχνά χρησιμοποιούμενη παράμετρος μέτρησης της απόδοσης αυτής είναι η *παράλληλη επιτάχυνση* (parallel speedup).

Ο παραδοσιακός ορισμός της επιτάχυνσης, ως ο λόγος του χρόνου της σειριακής εκτέλεσης, δια του χρόνου της παράλληλης εκτέλεσης, έχει γενικά γίνει αποδεκτός, τουλάχιστον σαν παραμετρική έννοια. Γενικά, όμως, η τάση της παραδοσιακής επιτάχυνσης είναι να μεροληπτεί υπέρ των *βραδύτερων* επεξεργαστών, ακόμη κι αν αγνοηθεί το κόστος επικοινωνίας και γι' αυτό θα πρέπει κανείς να κάνει προσεκτική χρήση αυτής της παραμέτρου, διότι η κατά κανόνα χρήση των ενδεικτικών τιμών της είναι δυνατόν να οδηγήσει σε λανθασμένα συμπεράσματα.

Άλλη ερμηνεία του ορισμού της επιτάχυνσης, η οποία συνήθως ταυτίζεται με τον παραδοσιακό ορισμό, χαρακτηρίζει τη λεγόμενη *σχετική* (reletive) επιτάχυνση (βλέπε Μπεκάκος, [72]), η οποία αναφέρεται στον ενυπάρχοντα παραλληλισμό (inherent parallelism) του παράλληλου αλγόριθμου υπό μελέτη και ορίζεται ως εξής:

$$S_p = \frac{\text{Χρόνος εκτέλεσης χρησιμοποιώντας έναν επεξεργαστή}}{\text{Χρόνος εκτέλεσης χρησιμοποιώντας p επεξεργαστές}} \qquad 1.8:1$$

Η αιτία που χρησιμοποιείται η σχετική επιτάχυνση, έγκειται στο ότι η απόδοση των παράλληλων αλγορίθμων διαφέρει ανάλογα με τον αριθμό των επεξεργαστών που διατίθενται. Συγκρίνοντας τη χρονική πολυπλοκότητα του ίδιου αλγόριθμου με έναν διαφορετικό αριθμό επεξεργαστών, είναι δυνατόν να προσδιοριστούν οι διακυμάνσεις (variations) και οι εξασθενήσεις (degradations) από την αρχή εφαρμογής του παραλληλισμού. Έτσι, ενώ η *απόλυτη* (absolute) επιτάχυνση (βλέπε Μπεκάκος, [72]) χρησιμοποιείται για την εκτίμηση των παράλληλων αλγορίθμων, η σχετική επιτάχυνση προτιμάται κυρίως για τη μελέτη της *εσωτερικής* απόδοσης αυτών. Απαραίτητη προυπόθεση όλων των παραπάνω είναι το γεγονός ότι δεν αναφερόμαστε καθόλου σε διανυσματικές αρχιτεκτονικές, οι οποίες μπορεί να θεωρηθεί ότι παρέχουν κάποια ειδική μορφή παράλληλης εκτέλεσης.

Με βάση το πλαίσιο εφαρμογής των παραμετρικών εννοιών της επιτάχυνσης, τρία είναι τα βασικά μοντέλα επιτάχυνσης τα οποία κατά κύριο λόγο έχουν μελετηθεί. Αυτά αφορούν, αντίστοιχα, στην *επιτάχυνση καθορισμένης-διάστασης* (fixed-size speedup), στην *επιτάχυνση καθορισμένου-χρόνου* (fixed-time speedup), και στην *επιτάχυνση περιορισμένης μνήμης* (memory-bounded speedup) (βλέπε Μπεκάκος, [72]).

Στο πρώτο μοντέλο έρευνας της επιτάχυνσης, καθορίζονται οι διαστάσεις του προβλήματος, ενώ με τη συνεχή αύξηση της διαθέσιμης υπολογιστικής ισχύος επιτυγχάνεται μία συνεχής μείωση της χρονικής πολυπλοκότητας του προβλήματος. Στο δεύτερο μοντέλο, όταν αυξάνεται η διαθέσιμη υπολογιστική ισχύς, αυξάνονται και οι διαστάσεις του προς επίλυση προβλήματος, εκτελούνται περισσότερες λειτουργίες και επιδιώκεται μία ακριβέστερη λύση του προβλήματος, ενώ διατηρείται αμετάβλητη η συνολική χρονική πολυπλοκότητα. Στο τρίτο μοντέλο της επιτάχυνσης περιορισμένης μνήμης, όπως και στο δεύτερο μοντέλο, έχουμε μία κλιμάκωση των διαστάσεων του προβλήματος με τον αριθμό των διαθέσιμων επεξεργαστών. Η διαφορά μεταξύ των δύο τελευταίων μοντέλων έγκειται στο ότι, στην περίπτωση της επιτάχυνσης περιορισμένης μνήμης, η χωρητικότητα της μνήμης είναι ο επικρατέστερος περιοριστικός παράγοντας της κλιμάκωσης. Η παράμετρος *χρόνος εκτέλεσης* μπορεί να πάρει διάφορες τιμές στο τελευταίο μοντέλο.

Οι παράλληλοι αλγόριθμοι συχνά συγκρίνονται κάτω από διαφορετικές *επιβαρύνσεις* ως προς τον προγραμματισμό, ενώ πολλές φορές συγκρίνονται

έμμεσα χρησιμοποιώντας αποτελέσματα από διαφορετικές μηχανές. Οι πρακτικές αυτές προσεγγίσεις εγείρουν μερικές ερωτήσεις, όπως : *Επιτυγχάνει ο ίδιος αλγόριθμος την ίδια απόδοση σε διαφορετικά συστήματα υπολογιστών; Επιτυγχάνει ο βελτιστοποιημένος κώδικας, ο οποίος δίνει το μικρότερο χρόνο εκτέλεσης, την ίδια ή καλύτερη απόδοση απ' ότι ο μη βελτιστοποιημένος κώδικας;*

Για να μπορεί να θεωρηθεί μία παράμετρος απόδοσης ως *δίκαιη* (fair), θα πρέπει ο ίδιος αλγόριθμος να επιτυγχάνει την ίδια ή καλύτερη απόδοση σε μία *'ταχύτερη'* μηχανή, και ο βελτιστοποιημένος κώδικας να επιτυγχάνει την ίδια ή καλύτερη απόδοση απ' ότι ο μη βελτιστοποιημένος κώδικας.

Ορισμός 1.8.1 : Μία παράμετρος απόδοσης είναι *ανεξάρτητη* από τη μηχανή (machine-independent), αν η απόδοση για ένα δεδομένο αλγόριθμο είναι η ίδια σε κάθε σύστημα υπολογιστή, όταν το κόστος επικοινωνίας είναι αμελητέο.

Ορισμός 1.8.2 : Μία παράμετρος απόδοσης είναι *ανεξάρτητη* του προγραμματισμού, αν η απόδοση είναι ανεξάρτητη της προσπάθειας προγραμματισμού (programming effort), όταν το κόστος επικοινωνίας είναι αμελητέο.

Έχει διαπιστωθεί ότι η *σχετική* επιτάχυνση είναι ανεξάρτητη της μηχανής, εάν και μόνον εάν είναι ανεξάρτητη του προγραμματισμού. Η επιτάχυνση, όμως, η πιο συνηθισμένη παράμετρος απόδοσης που χρησιμοποιείται, είναι εξαρτημένη και από τη μηχανή, αλλά και από τον προγραμματισμό. Κατά συνέπεια, χρειάζονται καινούργιες, καλύτερα ορισμένες παράμετροι απόδοσης, οι οποίες θα παρέχουν μία *δίκαιη*, και επομένως αξιόπιστη, ενδεικτική μέτρηση της απόδοσης. Η ιδέα της *κλιμακωτής* (scaled) επιτάχυνσης (βλέπε Μπεκάκος, [72]) επιτρέπει την αντίστοιχη αύξηση των διαστάσεων του προβλήματος με την αύξηση της υπολογιστικής ισχύος και δίνει έμφαση στο πόση εργασία *διεκπεραιώνεται* σε ένα καθορισμένο χρονικό διάστημα.

Ένα αναμενόμενο βήμα στην έρευνα για μία καλύτερη παράμετρο μπορεί να αποτελεί η εννοιολογική προέκταση της ιδέας της *κλιμάκωσης*. Το γεγονός αυτό οδήγησε στον ορισμό μιας νέας παραμέτρου, της *επιδιάστασης* (sizeup) (βλέπε Μπεκάκος, [72]):

$$\text{Επιδιάσταση} = \frac{\text{Παράλληλη εργασία (Parallel work)}}{\text{Σειριακή Εργασία (Sequential Work)}} \qquad 1.8:2$$

Σε μια μέτρηση της επιδιάστασης, η διάσταση του προβλήματος κλιμακώνεται, έτσι ώστε ο παράλληλος χρόνος εκτέλεσης να διατηρείται σταθερός, καθώς ο αριθμός των επεξεργαστών αυξάνεται. Η επιδιάσταση είναι ανεξάρτητη τόσο της μηχανής, όσο και του προγραμματισμού, στο βαθμό που η σειριακή εργασία δεν αυξάνεται ανάλογα με τη διάσταση του προβλήματος, διαφορετικά μεροληπτεί και αυτή υπέρ των βραδύτερων επεξεργαστών ή των 'άσχημα' κωδικοποιημένων προγραμμάτων. Αυτή η εξάρτηση, όμως, είναι πολύ μικρότερη στην περίπτωση της επιδιάστασης απ' ότι είναι στην περίπτωση της παραδοσιακής επιτάχυνσης. Κατά συνέπεια, η παράμετρος επιδιάσταση είναι δικαιότερη απ' ότι η παράμετρος παραδοσιακή επιτάχυνση, θα πρέπει, όμως, να υπογραμμιστεί το γεγονός ότι η επιδιάσταση στηρίζεται στην αρχή της κλιμακωτής επιτάχυνσης και παρέχει μία δίκαιη μέτρηση κάτω από συγκεκριμένες μόνον συνθήκες.

Από τα παραπάνω καθίσταται προφανές ότι η πρόκληση για τον προσδιορισμό μιας καλύτερης και ελεύθερης περιορισμών, απόλυτα δίκαιης, παραμέτρου αποτίμησης απόδοσης, για καθορισμένης διάστασης προβλήματα, παραμένει.

1.9 Παράγοντες Περιορισμού της Παράλληλης Απόδοσης

Στην Παράγραφο αυτή θα παρουσιαστούν μερικές από τις πλέον σημαντικές πηγές μείωσης της απόδοσης ενός παράλληλου προγράμματος, το οποίο υλοποιείται σε ένα πραγματικό σύστημα πολλαπλών επεξεργαστών.

1. Ανταγωνισμός για Κατοχή της Μνήμης (Memory Contention)

Αναφέρεται στην καθυστέρηση της εκτέλεσης που οφείλεται στην αναμονή των επεξεργαστών για να αποκτήσουν πρόσβαση σε κάποια θέση της μνήμης, η οποία χρησιμοποιείται τη δεδομένη χρονική στιγμή από κάποιον άλλον επεξεργαστή. Το πρόβλημα αυτό είναι δυνατόν να προκύψει όταν τα δεδομένα προσπελάζονται από ένα μεγάλο πλήθος επεξεργαστών και αφορά μόνον στα συστήματα καταμεριζόμενης μνήμης, δηλαδή τα συστήματα πολυεπεξεργασίας.

2. Χρήση Εκτεταμένου Σειριακού Κώδικα

Σε οποιονδήποτε παράλληλο αλγόριθμο υπάρχουν πάντοτε κομάτια καθαρά σειριακού κώδικα, στα οποία εκτελούνται βασικές λειτουργίες όπως είναι η

αρχικοποίηση των μεταβλητών. Σε μερικούς αλγόριθμους, αυτός ο σειριακός κώδικας μπορεί να περιορίζει σημαντικά τη συνολική επιτάχυνση που επιτυγχάνεται.

3. Χρόνος Δημιουργίας Διεργασιών

Σε οποιοδήποτε πραγματικό σύστημα, η δημιουργία των παράλληλων διεργασιών απαιτεί ένα συγκεκριμένο ποσοστό του συνολικού χρόνου εκτέλεσης. Στο βαθμό που οι δημιουργούμενες διεργασίες είναι *'μικρής διάρκειας'*, η χρονική επιβάρυνση που προκαλεί η δημιουργία τους είναι πολύ μεγαλύτερη από το χρόνο που εξοικονομείται από την εφαρμογή της έννοιας του παραλληλισμού στους αλγόριθμους αυτούς.

4. Επικοινωνιακή Καθυστέρηση

Η επιβάρυνση αυτή αναφέρεται στις καθυστερήσεις που εισάγει η ανταλλαγή δεδομένων μεταξύ των επεξεργαστών και αφορά μόνο στα συστήματα πολυ-υπολογιστών. Σε μερικές περιπτώσεις η επικοινωνία δύο επεξεργαστών μπορεί να προυποθέτει την προώθηση των αντίστοιχων μηνυμάτων τους κατά μήκος μιας μεγάλης διαδρομής από γραμμές επικοινωνίας που διέρχονται από άλλους επεξεργαστές. Οι προκύπτουσες καθυστερήσεις μπορεί να είναι αποφασιστικής σημασίας για την απόδοση του παράλληλου αλγόριθμου.

5. Καθυστέρηση Συγχρονισμού

Ο συγχρονισμός των παράλληλων διεργασιών εισάγει επίσης χρονικές καθυστερήσεις, οι οποίες οφείλονται στο ότι οι διεργασίες αναγκάζονται να περιμένουν η μία την άλλη σε συγκεκριμένα σημεία του προγράμματος. Κάτι τέτοιο είναι δυνατόν να επιφέρει σημαντική μείωση στην επιτάχυνση των αλγορίθμων.

6. Μη Ισορροπημένη Κατανομή του Φόρτου Εργασίας

Στο βαθμό που τα υπολογιστικά έργα ενός αλγόριθμου παράγονται δυναμικά και με εντελώς απρόβλεπτο τρόπο, θα πρέπει αυτά να εκχωρούνται επίσης δυναμικά στους διάφορους επεξεργαστές. Κάτι τέτοιο, συχνά, συνεπάγεται το γεγονός ότι ενώ μερικοί επεξεργαστές μπορεί να είναι αδρανείς, άλλοι θα πρέπει να εκτελέσουν περισσότερα υπολογιστικά έργα από όσα μπορούν να χειριστούν.

Κεφάλαιο 2

Μορφές Παράλληλης Επεξεργασίας και Αλγοριθμο-δομημένες Αρχιτεκτονικές

2.1 Εισαγωγή

Η εξέλιξη στην τεχνολογία των ημιαγωγών οδήγησε στην εναλλακτική λύση της χρήσης του πυριτίου (silicon) και του μετάλλου-οξειδίου ημιαγωγού (Metal Oxide Semiconductor - MOS), το οποίο αν και ήταν 5 εως 10 φορές πιο αργό, προσέφερε τις δυνατότητες για πολύ υψηλότερου επιπέδου πυκνότητες ολοκλήρωσης. Έτσι, στις αρχές της δεκαετίας του '80 η κατασκευή των μικροεπεξεργαστών άρχισε, με ταχύτητες και χωρητικότητες παρόμοιες με αυτές των υπολογιστών της πρώτης γενιάς, αλλά σε διαστάσεις ολίγων μόνον χιλιοστών του μέτρου ενός τσιπ από πυρίτιο (silicon chip). Το γεγονός αυτό, σε συνδυασμό με τις τεχνολογικές βελτιώσεις που αφορούσαν στις τεχνικές μικρογράφησης (miniaturization techniques), αποτέλεσε τον πλέον αποφασιστικό παράγοντα για τη στροφή και την επικέντρωση της έρευνας στο χώρο των VLSI επεξεργαστών και της *συνδρομικής* (concurrent) επεξεργασίας.

Ένας νέος θεωρητικός τομέας αναπτύχθηκε, ο οποίος αφορούσε στην έρευνα υπολογιστικών μοντέλων που περιελάμβαναν τα χαρακτηριστικά των VLSI αρχιτεκτονικών. Σε αυτά τα υπολογιστικά μοντέλα δεν αγνοούνταν πλέον οι φυσικοί περιορισμοί που επιβάλλονταν από το κόστος των γραμμών επικοινωνίας

όπως συνέβαινε στα αντίστοιχα μοντέλα των συμβατικών υπολογιστών. Έτσι, επήλθε μία μετάβαση από τα μοντέλα στα οποία οι λογικές λειτουργίες αποτελούσαν τον κεντρικό πυρήνα, στα μοντέλα στα οποία η γεωμετρία των γραμμών επικοινωνίας ήταν εξίσου σημαντική. Η δυνατότητα της εύκολης υλοποίησης κάνοντας χρήση της VLSI τεχνολογίας, τόσο των στοιχείων επεξεργασίας, όσο και των στοιχείων μνήμης, ενθάρρυνε τη δημιουργία υπολογιστικών δομών με μεγάλο βαθμό συνδρομικότητας στις οποίες ήταν δυνατό να εκτελείται ταυτόχρονα ένα μεγάλο πλήθος πράξεων. Το βασικό χαρακτηριστικό των υπολογιστικών αυτών δομών υπήρξε η *κανονικότητα* (regularity) της γεωμετρίας τους. Μολονότι, όμως, ήταν δυνατός ο σχεδιασμός τέτοιων VLSI υπολογιστικών δομών, η πολυμορφία των εφαρμογών, η οποία επέβαλε διαφορετικά είδη συνδρομικής επεξεργασίας, έθετε μερικούς πολύ σημαντικούς προβληματισμούς, όπως : *Με ποιο τρόπο είναι δυνατόν να χρησιμοποιηθούν αποδοτικά οι υπολογιστικές αυτές δομές; Υπάρχουν κάποιες γενικές αρχές ή θεωρίες οι οποίες αφορούν στο σχεδιασμό υψηλού επιπέδου συνδρομικών συστημάτων;*

Από υπολογιστικής άποψης, το βασικό χαρακτηριστικό της VLSI τεχνολογίας είναι ότι δεν υπάρχει διαφορά μεταξύ του λογισμικού (αλγόριθμοι) και του υλικού (κυκλώματα), δηλαδή, κάθε αλγόριθμος είναι δυνατόν να υλοποιηθεί από ένα κατάλληλο κύκλωμα. Από πρακτικής άποψης, κάτι τέτοιο επιτρέπει την άμεση ολοκλήρωση σύνθετων λειτουργιών, ενώ από θεωρητικής απόψης, παρέχεται μία δυνατότητα εκτίμησης του υλικού ως προς τον απαιτούμενο φυσικό χώρο που καταλαμβάνει το εκάστοτε κύκλωμα. Η εκτίμηση αυτή αποβλέπει, συνήθως, στη σχεδίαση χωρο-αποδοτικών κυκλωμάτων και χρησιμοποιείται σε κάθε προσπάθεια ανάλυσης της υπολογιστικής τους πολυπλοκότητας. Δύο είναι οι βασικές προσεγγίσεις που αφορούν σε μία τέτοια ανάλυση της υπολογιστικής πολυπλοκότητας. Η πρώτη αφορά στη δημιουργία εξειδικευμένων αρχιτεκτονικών για την αποδοτική και πιθανώς βέλτιστη επίλυση ενός δεδομένου προβλήματος και η δεύτερη αφορά στη δημιουργία ευφυών (versatile) αρχιτεκτονικών οι οποίες είναι κατάλληλες για την επίλυση διαφόρων υπολογιστικών προβλημάτων παρέχοντας μία καλή, αν όχι βέλτιστη απόδοση. Ένα παράδειγμα μιας τέτοιας ευφυούς αρχιτεκτονικής είναι το γράφημα ανάμειξης-ανταλλαγής, το οποίο είναι κατάλληλο για την υλοποίηση αλγορίθμων ταχείας ταξινόμησης, αλγορίθμων οι οποίοι πραγματοποιούν πολυωνυμικούς υπολογισμούς, κ.ά.

2.2 Μορφές Παράλληλης Επεξεργασίας

Γενικά, η απόδοση των συστημάτων με έναν επεξεργαστή είναι σχετικά περιορισμένη, παρόλο που είναι δυνατόν η έννοια του παραλληλισμού να εφαρμοστεί τόσο στο υλικό, όσο και στο λογισμικό του συστήματος, υιοθετώντας διάφορους μηχανισμούς, όπως είναι η πολλαπλότητα των λειτουργικών μονάδων, ο παραλληλισμός και η τεχνική της σωλήνωσης εντός της ΚΜΕ, η επικάλυψη των λειτουργιών της ΚΜΕ και των μονάδων εισόδου/εξόδου, η χρήση ενός ιεραρχικού συστήματος μνήμης, η εξισορρόπηση του εύρους ζώνης των υποσυστημάτων και ο πολυπρογραμματισμός και χρονικός καταμερισμός.

Οι περιορισμένες δυνατότητες αύξησης της υπολογιστικής ισχύος στους σειριακούς υπολογιστές καθιστούν τη χρήση των παράλληλων υπολογιστών ως την πλέον υποσχόμενη λύση για την επίτευξη του παραπάνω στόχου. Επιπλέον, είναι γενικά αποδεκτό το γεγονός ότι η τεχνολογία προηγείται σημαντικά του λογισμικού και της αρχιτεκτονικής, γεγονός που οφείλεται στο ότι τα σύγχρονα ηλεκτρονικά δομοστοιχεία είναι πολύ αξιόπιστα, σε σχέση με τα πολύπλοκα συστήματα του λογισμικού, τα οποία αναπόφευκτα εισάγουν λάθη στα διάφορα προγράμματα, αφού υπόκεινται σε συνεχή συντήρηση (maintenance), τροποποίηση (modification) και εξέλιξη (development).

2.2.1 Διανυσματικές Εντολές (Vector Instructions)

Μία διανυσματική εντολή μήκους n, ορίζει ένα πλήθος από n πανομοιότυπες εκτελέσεις της ίδιας εντολής, όπου οι τελεστέοι για κάθε εκτέλεση δεν εξαρτώνται από το αποτέλεσμα κάποιας άλλης εκτέλεσης της εντολής. Το γεγονός αυτό επιτρέπει την παράλληλη ή ταυτόχρονη υλοποίηση όλων των εκτελέσεων της εντολής.

Συνήθως, μία διανυσματική εντολή είναι σύνθετη και αποτελείται από ένα πλήθος *στοιχειωδών* υποεντολών, που για την εκτέλεση της καθεμίας από αυτές απαιτείται ο ίδιος στοιχειώδης χρόνος. Στο βαθμό που κάτι τέτοιο δεν ισχύει, τότε ως χρόνος εκτέλεσης της κάθε υποεντολής θεωρείται ο μεγαλύτερος χρόνος εκτέλεσης που απαιτείται για κάποια από τις υποεντολές.

Υπάρχουν διάφορα επίπεδα διαχωρισμού μιας εντολής σε υποεντολές. Χαρακτηριστικό παράδειγμα είναι ο διαχωρισμός μιας απλής εντολής, που για έναν χρήστη φαίνεται σαν κάτι το *συμπαγές*, σε τέσσερις φάσεις:

- *Υποεντολή 1*: Προσκόμιση της εντολής από τη μνήμη (Instruction Fetch - IF)
- *Υποεντολή 2*: Αποκωδικοποίηση της εντολής (Instruction Decode - ID)
- *Υποεντολή 3*: Προσκόμιση των τελεστέων από τη μνήμη (Operand Fetch - OF)
- *Υποεντολή 4*: Εκτέλεση της πράξης (Execution - EXEC)

Αν θεωρηθεί μία διανυσματική εντολή μήκους n, η οποία υποδιαιρείται σε p υποεντολές με μέγιστο χρόνο εκτέλεσης κάποιας από τις υποεντολές ίσο με t, τότε ο συνολικός χρόνος εκτέλεσης της εντολής ισούται με

$$T_{vec} = p * t, \qquad (2.2.1:1)$$

ενώ ο ίδιος χρόνος σε έναν υπολογιστή von-Neumann ισούται με

$$T_{seq} = n * p * t, \qquad (2.2.1:2)$$

εξαιτίας της ακολουθιακής εκτέλεσης της διανυσματικής εντολής.

2.2.2 Σωληνοποίηση ή Αγωγοποίηση (Pipelining)

Η χρήση των σωληνωτών επεξεργαστών υιοθετήθηκε, μετά από την ανακάλυψη των τεχνικών του είδους γραμμής συναρμολόγησης, με σκοπό τη βελτίωση της απόδοσης της αριθμητικής μονάδας ή της μονάδας ελέγχου, με την αποσύνθεση μιας επαναλαμβανόμενης σειριακής διεργασίας σε υποδιεργασίες, οι οποίες είναι δυνατόν να εκτελεστούν από ειδικά αφιερωμένους για το σκοπό αυτό αυτόνομους (υπο)επεξεργαστές. Καθένας από αυτούς παραλαμβάνει τα στοιχεία πληροφορίας από τον προηγούμενό του (στη σειρά διασύνδεσης) και παραδίδει τα αποτελέσματά του στον αμέσως επόμενό του (στη σειρά διασύνδεσης) επεξεργαστή.

Κατ' αυτόν τον τρόπο, όλοι μαζί στη σειρά σχηματίζουν ένα σωλήνα (pipe), διαμέσω του οποίου ρέουν τα στοιχεία πληροφορίας. Ένας αγωγός μήκους p, μπορεί να εκτελέσει μια εντολή που είναι διαχωρισμένη σε p υποεντολές, όπως φαίνεται στο σχήμα 2.2.2-1. Τα X_i, όπου i = 1,2, ..., p, συμβολίζουν και την υποεντολή, αλλά και

τον αντίστοιχο επεξεργαστή, ενώ ο κάθε επεξεργαστής θεωρείται ότι εκτελεί την υποεντολή του σε χρόνο t.

Σχήμα 2.2.2-1: Ροή Στοιχείων Πληροφορίας στο Σωλήνα

Κατά συνέπεια, ο συνολικός χρόνος για την εκτέλεση μιας διανυσματικής εντολής ισούται με

$$T_{pipe} = t_{init} + (p - 1 + n) * t, \qquad (2.2.2:1)$$

όπου t_{init} ο χρόνος αρχικοποίησης του υλικού για την εκτέλεση της συγκεκριμένης εντολής, p το πλήθος των υποεντολών της εντολής και n το μήκος της προς εκτέλεση διανυσματικής εντολής.

2.2.3 Διανυσματικός Υπολογιστής ή Διανύσματα Επεξεργαστών (Vector Processor)

Ένα σύστημα αυτής της μορφής περιλαμβάνει ένα πλήθος από q στοιχεία επεξεργασίας (Processing Elements-PEs), η πολυπλοκότητα των οποίων κυμαίνεται από στοιχειώδεις επεξεργαστές, για παράδειγμα p αθροιστές για την περίπτωση πραγματικών προσθέσεων, μέχρι πλήρη υπολογιστικά συστήματα. Τα PEs αυτά έχουν τη δυνατότητα της ταυτόχρονης και ανεξάρτητης λειτουργίας. Κατ' αυτόν τον τρόπο μπορούν να λαμβάνουν χώρα q εκτελέσεις της ίδιας εντολής στον ίδιο χρόνο της μιας εκτέλεσης.

Ο συνολικός χρόνος εκτέλεσης μιας διανυσματικής εντολής μήκους n, ισούται με

$$T_{tot} = \left\lceil \frac{n}{q} \right\rceil * p * t \qquad (2.2.3:1)$$

Ενώ η σωληνοποίηση θεωρείται μία μορφή παραλληλισμού *'ως προς το χρόνο'* και αυτό διότι δε χρησιμοποιούνται πολλοί επεξεργαστές, αλλά ένας, *διαχωρισμένος*

σε λίγους (υπο)επεξεργαστές, ένας διανυσματικός υπολογιστής χρησιμοποιεί q αντίγραφα του ίδιου επεξεργαστή και κατά συνέπεια υφίσταται μία μορφή παραλληλισμού *'ως προς το χώρο'*.

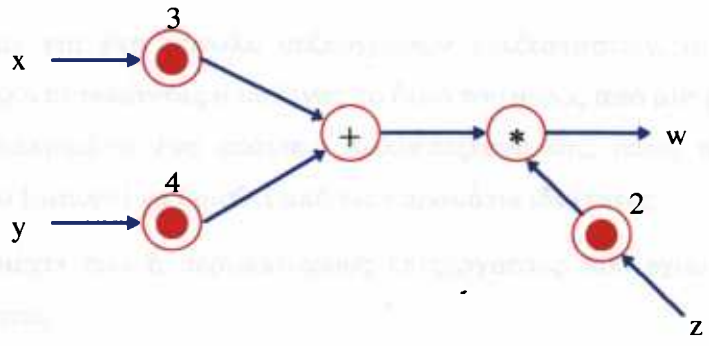
2.2.4 Μηχανές Ροής Στοιχείων Πληροφορίας (Data Flow Machines)

Η βασική αρχή του παραλληλισμού είναι η κάθε εκτέλεση να αρχίζει *αμέσως* μόλις τα απαραίτητα στοιχεία πληροφορίας εισόδου καταστούν *διαθέσιμα*. Σε αυτό ακριβώς το γεγονός στηρίζεται η λειτουργία των μηχανών ροής στοιχείων πληροφορίας. Με άλλα λόγια, δεν υπάρχει κάποιος μετρητής προγράμματος (program counter), αλλά η διαθεσιμότητα των στοιχείων πληροφορίας *πυροδοτεί* την έναρξη εκτέλεσης της εντολής ανεξάρτητα από τη σειρά που έχει η εντολή στο πρόγραμμα.

Κατά συνέπεια, αφού η εντολή αρχίζει να εκτελείται αμέσως μόλις αυτό καταστεί δυνατό, ο τρόπος λειτουργίας αυτός μας προσφέρει τις δυνατότητες για την επίτευξη του μέγιστου θεωρητικά δυνατού παραλληλισμού, με μόνο ουσιαστικό περιορισμό αυτόν που επιβάλλεται από το υλικό.

Η μοντελοποίηση (modeling) αυτού του είδους των παράλληλων μηχανών επιτυγχάνεται με κατάλληλα γραφήματα ροής δεδομένων. Στο σχήμα 2.2.4-1 περιγράφεται με ένα γράφημα ροής η εκτέλεση της πράξης:

$$w = (x + y) * z, \text{ όταν } x=3,\ y=4 \text{ και } z=2$$



Σχήμα 2.2.4-1: Γράφημα Ροής

2.2.5 Συστολικά Δίκτυα (Systolic Networks) και με Κυματοειδή Μορφή Επεξεργασίας Πίνακες Επεξεργαστών (Wavefront Array Processors)

Οι συστολικές διατάξεις (Kung, [48]), στις οποίες θα γίνει εκτενής αναφορά στις τελευταίες Παραγράφους του παρόντος Κεφαλαίου, συνδυάζουν δύο από τις προηγούμενα αναφερθείσες μορφές επεξεργασίας : Τη σωληνοποίηση και τη διανυσματική επεξεργασία. Αποτελούνται από ένα σύνολο στοιχειωδών, συνήθως, επεξεργαστών οι οποίοι λειτουργούν με πλήρως συγχρονισμένο τρόπο, ενώ η γενική ιδέα θυμίζει τον τρόπο λειτουργίας της καρδιάς, η οποία διακρίνεται σε δύο φάσεις, τη φάση της *συστολής* (systole) και τη φάση της *διαστολής* (diastole). Κατά τη φάση της συστολής πραγματοποιείται η είσοδος των στοιχείων σε κάθε επεξεργαστή, ενώ κατά τη φάση της διαστολής πραγματοποιείται η έξοδος των αποτελεσμάτων, ή η μεταβίβαση των στοιχείων πληροφορίας, συνήθως, στους αμέσως γειτονικούς επεξεργαστές.

Από την άλλη μεριά, οι πίνακες επεξεργαστών με κυματοειδή μορφή επεξεργασίας (WAP) (Bekakos, [13]), συνδυάζουν τη σωληνοποίηση με τον τρόπο λειτουργίας των μηχανών ροής στοιχείων πληροφορίας. Σε αυτές τις διατάξεις δεν υπάρχει καθολική μορφή συγχρονισμού, αλλά θεωρείται ότι τα δεδομένα διαδίδονται υπό μορφή κυμάτων κατά τη διαγώνια προέκταση της διάταξης. Επιπλέον, τα PEs θεωρείται ότι διαθέτουν μία δυνατότητα ασύγχρονης αναμονής η οποία χαρακτηρίζει κυρίως τις μηχανές ροής στοιχείων πληροφορίας.

2.2.6 Συστήματα Πολυεπεξεργασίας

Πρόκειται για ένα σύνολο συζευγμένων επεξεργαστών, οι οποίοι εργάζονται παράλληλα εκτελώντας ο καθένας το δικό του μέρος από μία μεγαλύτερη εργασία. Πιο συγκεκριμένα, ένα σύστημα πολυεπεξεργασίας, όπως ήδη αναφέρθηκε στο Κεφάλαιο 1, μπορεί να ορισθεί από τις παρακάτω ιδιότητες:

1. Περιέχει δύο ή περισσότερους επεξεργαστές που έχουν τις ίδιες περίπου δυνατότητες.

2. Όλοι οι επεξεργαστές διαμοιράζονται την προσπέλαση σε κάποια κοινή ή καταμεριζόμενη μνήμη, χωρίς, όμως, αυτό να είναι απαγορευτικός περιορισμός του να διαθέτουν και δικές τους, *ιδιωτικές*, μνήμες.

3. Όλοι οι επεξεργαστές διαμοιράζονται την προσπέλαση σε διαύλους I/O, μονάδες ελέγχου και λοιπές περιφερειακές συσκευές.

4. Το όλο σύμπλεγμα είναι *ενιαίο*, από την άποψη ότι διευθύνεται από ένα Λειτουργικό Σύστημα, το οποίο επιτρέπει την *αλληλεπίδραση* και την *επικοινωνία* μεταξύ των επεξεργαστών και μεταξύ των προγραμμάτων τους σε όλα τα επίπεδα : Από το συνολικό έργο μέχρι και ένα μεμονωμένο βήμα κάποιας υποδιεργασίας του έργου, από τα αρχεία πληροφοριών μέχρι και ένα διακεκριμένο στοιχείο πληροφορίας.

Το ενιαίο του συστήματος αποτελεί ένα από τα βασικά στοιχεία διάκρισης μεταξύ των συστημάτων πολυεπεξεργασίας και των κατανεμημένων συστημάτων.

2.3 Συστολικές Αρχιτεκτονικές

Η ανάπτυξη της VLSI τεχνολογίας κατέστησε εφικτή τη δημιουργία εξειδικευμένων περιφερειακών συσκευών χαμηλού κόστους για την άμεση επίλυση συγκεκριμένου τύπου προβλημάτων. Η δυνατότητα αυτή αποτέλεσε έναν αποφασιστικό παράγοντα στη στροφή προς την ανάπτυξη παράλληλων αλγορίθμων υψηλής απόδοσης, οι οποίοι είναι δυνατόν να υλοποιηθούν άμεσα σε τέτοιου είδους συσκευές υλικού (συστολικές διατάξεις). Ο όρος απόδοση, στη συγκεκριμένη περίπτωση, αναφέρεται, κυρίως, στη *ρυθμαπόδοση* (throughput) που επιτυγχάνεται όταν μία τέτοια εξειδικευμένη περιφερειακή συσκευή προσαρτάται σε έναν γενικού σκοπού κεντρικό υπολογιστή. Κάτι τέτοιο συνεπάγεται ότι λαμβάνονται υπόψη, τόσο ο απαιτούμενος χρόνος εκτέλεσης των εντολών, όσο και ο χρόνος που καταναλώνεται για την εισόδο/έξοδο, για την εφαρμογή κάποιας μορφής ελέγχου, καθώς και για την κίνηση των δεδομένων μέσα στη διάταξη.

Μία συστολική διάταξη αποτελεί έναν συνδυασμό ενός αγωγού και μιας διανυσματικής μηχανής. Διαθέτει πολλούς πανομοιότυπους *στοιχειώδεις* επεξεργαστές, οι οποίοι ακριβέστερα χαρακτηρίζονται ως Επεξεργαστές Βήματος Εσωτερικού Γινομένου (Inner Product Step Processors-IPSPs) και, κατά συνέπεια, υπάρχει μία συμπεριφορά παρόμοια με αυτήν του διανυσματικού υπολογιστή. Ταυτόχρονα, όμως, λαμβάνει χώρα και μία σωληνική μορφή επεξεργασίας μεταξύ των IPSPs. Η λειτουργία των IPSPs είναι απόλυτα συγχρονισμένη κάτω από τον έλεγχο ενός ρολογιού, στους κτύπους του οποίου μεταβιβάζουν/δέχονται στοιχεία

πληροφορίας (νέα ή ενδιάμεσα μερικά αποτελέσματα), ενώ η κύρια λειτουργία τους λαμβάνει χώρα στα μεταξύ των τακτικών τύπων διαστήματα. Τα IPSPs πρέπει να είναι *απλά* στην κατασκευή τους ώστε να είναι δυνατή η σχετικά φτηνή μαζική παραγωγή τους. Το όλο σύστημα, το οποίο θα πρέπει, και αυτό είναι βασική προυπόθεση της καλής λειτουργίας του, να έχει κανονική δομή, προσαρτάται σε έναν κεντρικό υπολογιστή ο οποίος αναλαμβάνει τη διακίνηση των στοιχείων πληροφορίας από και προς την εξειδικευμένη συστολική διάταξη.

Η χρήση της VLSI τεχνολογίας κατέστησε προφανές ότι οι *απλές* και *κανονικές* δομές διασύνδεσης των επεξεργαστών οδηγούν σε χαμηλού κόστους υλοποιήσεις με μεγάλη πυκνότητα, γεγονός το οποίο συνεπάγεται υψηλή απόδοση και ταυτόχρονα χαμηλή επιβάρυνση ως προς το κόστος κατασκευής των IPSPs. Για τους λόγους αυτούς, το ενδιαφέρον επικεντρώθηκε στην ανάπτυξη παράλληλων αλγορίθμων με κανονική ροή δεδομένων. Οι πλέον αντιπροσωπευτικοί αλγόριθμοι που χρησιμοποιήθηκαν για την επεξήγηση της λειτουργίας των συστολικών διατάξεων αφορούσαν στον πολλαπλασιασμό πίνακα με διάνυσμα, στον πολλαπλασιασμό πινάκων, στην LU-παραγοντοποίηση ενός πίνακα, κ.ά.

Ο απλούστερος τρόπος υλοποίησης ενός συστολικού αλγόριθμου είναι μέσω της κατασκευής μιας *εξειδικευμένης συστολικής διάταξης*, ακριβώς για τις απαιτήσεις του συγκεκριμένου αλγόριθμου. Η προσέγγιση αυτή φαίνεται να είναι λογική, αν ισχύει μία ή περισσότερες από τις ακόλουθες συνθήκες:

(I) Η απόδοση της συστολικής αρχιτεκτονικής είναι υψίστης σπουδαιότητας και η χρήση της είναι πλήρως κατανοητή.

(II) Η συστολική αυτή αρχιτεκτονική θα παραχθεί σε μεγάλες ποσότητες, παρά το γεγονός ότι αφορά μία συγκεκριμένη εφαρμογή.

(III) Η σχεδίαση και το κόστος της υλοποίησής της είναι χαμηλό, κάτι το οποίο στην πραγματικότητα συμβαίνει μόνο σε εκείνες τις συστολικές διατάξεις οι οποίες αποτελούνται από μερικούς τύπους μόνο πολύ απλών κελιών.

Οι *συστολικές αρχιτεκτονικές πολλών εφαρμογών* αποβλέπουν στην υλοποίηση ενός προκαθορισμένου συνόλου συστολικών αλγορίθμων. Η προσέγγιση αυτή βασίζεται στην παρατήρηση ότι πολλοί συστολικοί αλγόριθμοι μπορούν να εκτελεστούν σε συστολικές διατάξεις παρόμοιας δομής. Κατά συνέπεια, τέτοιου είδους συστολικοί επεξεργαστές, με ελάχιστες επιβαρύνσεις για την παροχή της

απαιτούμενης ευελιξίας, μπορούν να εκτελέσουν ένα πλήθος λειτουργιών κάτω από κάποιον έλεγχο.

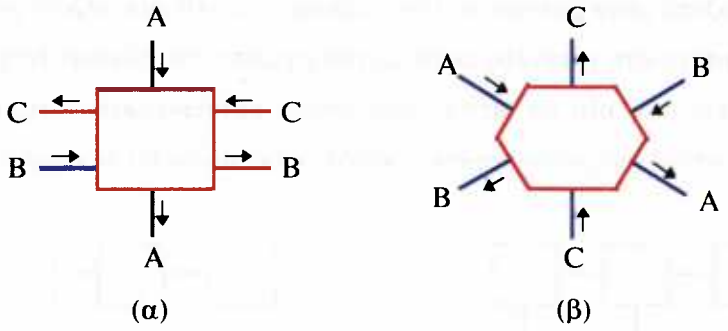
2.3.1 Βασικά Στοιχεία και Δομές Συστολικών Διατάξεων

2.3.1.1 Επεξεργαστής Βήματος Εσωτερικού Γινομένου

Η πιο κοινή λειτουργία, στους αλγόριθμους οι οποίοι θα παρουσιαστούν στις επόμενες Παραγράφους, είναι το ονομαζόμενο βήμα εσωτερικού γινομένου (IPS),

$$C \leftarrow C + A*B \qquad (2.3.1.1:1)$$

Ας θεωρήσουμε έναν επεξεργαστή ο οποίος διαθέτει τρεις καταχωρητές R_A, R_B και R_C. Ο κάθε καταχωρητής διαθέτει δύο γραμμές επικοινωνίας, εισόδου και εξόδου, αντίστοιχα. Στο σχήμα 2.3.1.1-1 παρουσιάζονται οι δύο τύποι γεωμετρίας για έναν τέτοιο επεξεργαστή.



Σχήμα 2.3.1.1-1: Γεωμετρίες για τον Επεξεργαστή Βήματος Εσωτερικού Γινομένου

Ορίζοντας μία βασική χρονική μονάδα σε συνάρτηση με τη λειτουργία του επεξεργαστή ισχύει, ότι σε κάθε μοναδιαίο χρονικό διάστημα ο επεξεργαστής μετατοπίζει (shifts) τα δεδομένα που βρίσκονται στις γραμμές εισόδου του, και οι οποίες υποδηλώνονται με τα γράμματα A, B και C, στους καταχωρητές R_A, R_B και R_C, αντίστοιχα, εκτελεί τη λειτουργία

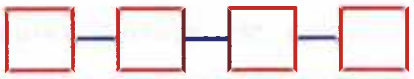
$$R_C \leftarrow R_C + R_A * R_B \qquad (2.3.1.1:2)$$

και καθιστά διαθέσιμες σαν δεδομένα εξόδου, στις γραμμές εξόδου του που υποδηλώνονται με A, B και C, αντίστοιχα, τις τιμές εισόδου για τους καταχωρητές R_A και R_B, μαζί με τη νέα τιμή του R_C.

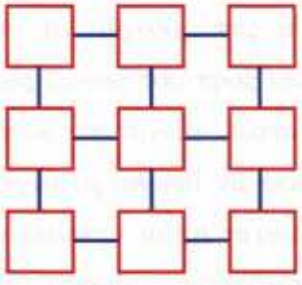
Όλα τα δεδομένα εξόδου είναι *ασφαλή* (latched) και η λογική της διάταξης υιοθετεί τους παλμούς χρονισμού, έτσι ώστε όταν ένας επεξεργαστής είναι συνδεδεμένος με κάποιον άλλο, η αλλαγή στα δεδομένα εξόδου του ενός, κατά τη διάρκεια ενός μοναδιαίου χρονικού διαστήματος, δεν επηρεάζει τα δεδομένα εισόδου στον άλλο επεξεργαστή κατά τη διάρκεια του ίδιου χρονικού διαστήματος. Το στοιχείο επεξεργασίας αυτής της μορφής, δεν είναι το μοναδικό που χρησιμοποιείται στις συστολικές αρχιτεκτονικές, αποτελεί όμως το βασικό στοιχείο υπολογισμών σε αυτές.

2.3.1.2 Συστολικές Διατάξεις

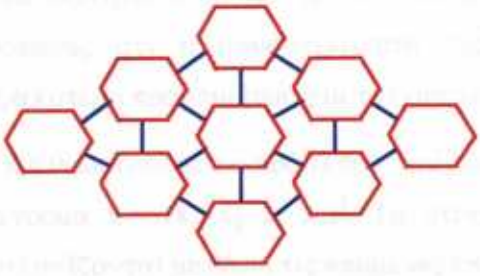
Μία συστολική διάταξη τυπικά αποτελείται από πολλούς, συνδεδεμένους μεταξύ τους, IPSPs. Η βασική οργάνωση της διάταξης που θα χρησιμοποιηθεί στις επόμενες Παραγράφους, για την παρουσίαση των συστολικών αλγορίθμων, είναι πλεγματοειδούς ζεύξης και όλες οι γραμμές επικοινωνίας ενός επεξεργαστή είναι με τους άμεσα γειτονικούς του επεξεργαστές, όπως φαίνεται στο σχήμα 2.3.1.2-1. Εάν διαγώνιες γραμμές επικοινωνίας προστεθούν κατά τη μία κατεύθυνση μόνο, η διάταξη η οποία προκύπτει ονομάζεται εξαγωνικο-συνδεδεμένη διάταξη.



(α): Γραμμική Διάταξη



(β): Πλεγματοειδής Διάταξη



(γ): Εξαγωνικο-συνδεδεμένη Διάταξη

Σχήμα 2.3.1.2-1: Συστολικές Διατάξεις

Οι επεξεργαστές οι οποίοι βρίσκονται στις περατωτικές θέσεις της συστολικής διάταξης μπορούν να έχουν εξωτερικές συνδέσεις με τη μνήμη του κεντρικού συστήματος. Έτσι, ένας διάδρομος δεδομένων εισόδου/εξόδου ενός περατωτικού επεξεργαστή μπορεί μερικές φορές να υποδηλωθεί ως μία γραμμή επικοινωνίας εισόδου/εξόδου για τη συστολική διάταξη. Ένας περάτωτικός επεξεργαστής μπορεί να δεχθεί δεδομένα από τη μνήμη του κεντρικού υπολογιστή, διαμέσω μιας τέτοιας εξωτερικής γραμμής επικοινωνίας, ή μπορεί να δεχθεί μία σταθερή τιμή, όπως η τιμή μηδέν. Από την άλλη μεριά, ένας περατωτικός επεξεργαστής μπορεί να στέλνει δεδομένα προς την κεντρική μνήμη διαμέσω μιας τέτοιας εξωτερικής γραμμής επικοινωνίας. Η έξοδος ενός περατωτικού επεξεργαστή μπορεί μερικές φορές να αγνοείται, γεγονός το οποίο υποδηλώνεται με την παράλειψη της αντίστοιχης γραμμής εξόδου του στη συστολική διάταξη.

2.3.2 Αλγόριθμος Πολλαπλασιασμού Πίνακα με Διάνυσμα

Η μονοδιάστατη γραμμική διάταξη αποτελεί την απλούστερη και πιο θεμελιώδη γεωμετρία διασύνδεσης επεξεργαστών. Οι καταχωρητές ολίσθησης (shift registers) μπορούν απευθείας να υλοποιήσουν τις γραμμικές διατάξεις.

Ο αλγόριθμος πολλαπλασιασμού πίνακα με διάνυσμα χρησιμοποιεί μία τέτοια γραμμική διάταξη. Θα πρέπει να σημειωθεί ότι το μέγεθος της συστολικής διάταξης, κάθε φορά, εξαρτάται μόνον από το εύρος ζώνης του προς επεξεργασία ταινιωτού πίνακα, ενώ είναι ανεξάρτητο από το μήκος της ταινίας. Κατά συνέπεια, μία συστολική διάταξη επεξεργαστών σταθερού μεγέθους μπορεί να επεξεργαστεί κατά σωληνωτό τρόπο, ταινιωτούς πίνακες με αυθαίρετα μήκη ταινιών. Είναι προφανές ότι η αποδοτικότητα της σωληνικής μορφής επεξεργασίας είναι μεγαλύτερη προκειμένου για ταινιωτούς πίνακες μεγάλου μήκους.

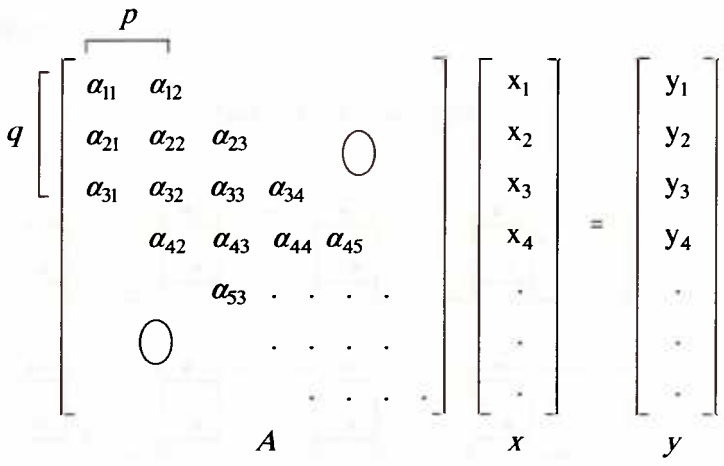
Ας θεωρήσουμε το πρόβλημα του πολλαπλασιασμού ενός πίνακα $A = (a_{ij})$ με ένα διάνυσμα $x = (x_1, x_2, ..., x_n)^T$. Τα στοιχεία του n-διανύσματος $y = (y_1, y_2, ..., y_n)^T$ υπολογίζονται με βάση τις επόμενες επαναληπτικές σχέσεις

$$y_i^{(1)} = 0$$

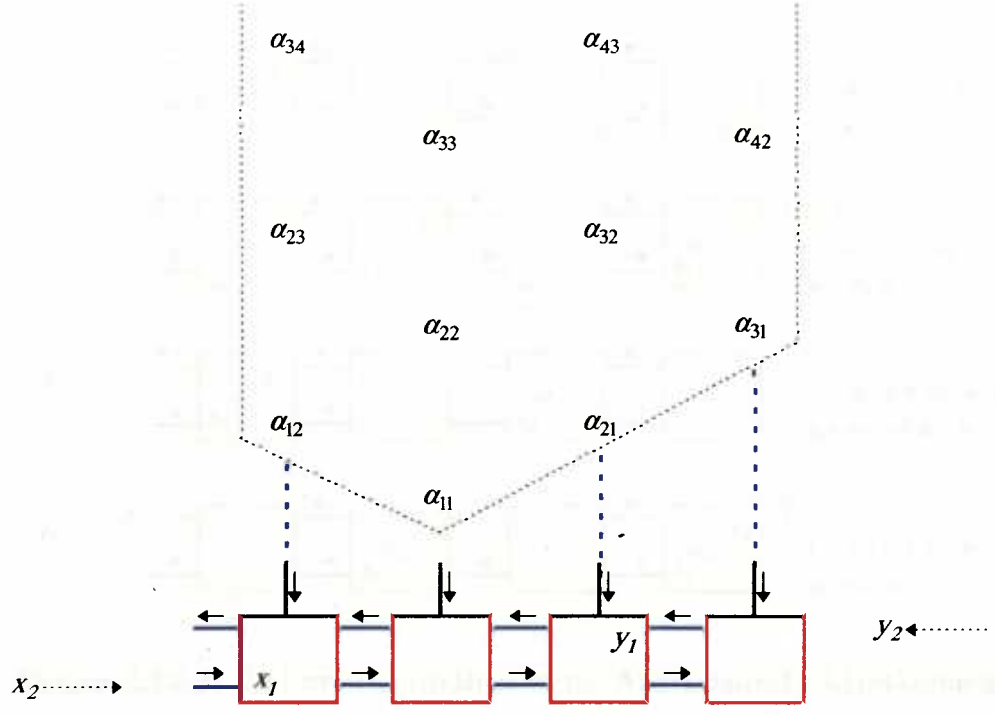
$$y_i^{(k+1)} = y_i^{(k)} + a_{ik} * x_k \qquad (2.3.2:1)$$

$$y_i = y_i^{(n+1)}$$

Οι επαναληπτικές σχέσεις (2.3.2:1) μπορούν να υπολογιστούν με τη μετατόπιση των x_i και y_i διαμέσω w=p+q-1 γραμμικά συνδεδεμένων επεξεργαστών, χρησιμοποιώντας την τεχνική της σωλήνωσης. Ο αλγόριθμος αυτός, για την περίπτωση που το p=2 και το q=3, καθώς και η συστολική διάταξη, η οποία στη συγκεκριμένη περίπτωση θα αποτελείται από τέσσερις γραμμικά συνδεδεμένους επεξεργαστές, περιγράφονται στα σχήματα 2.3.2-1 και 2.3.2-2, αντίστοιχα.

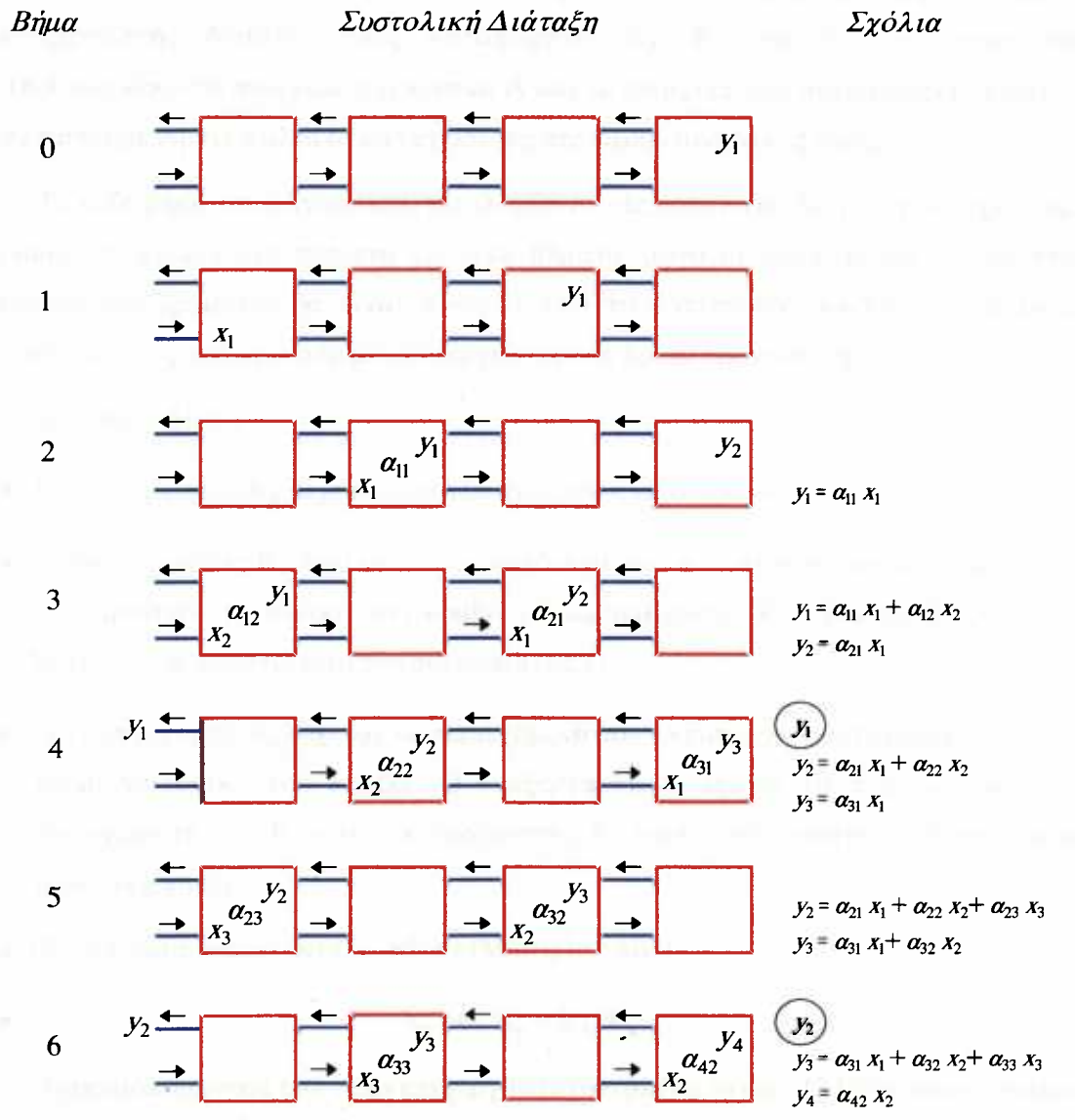


Σχήμα 2.3.2-1: Πολλαπλασιασμός Ταινιωτού Πίνακα με Διάνυσμα (για p=2 και q=3)



Σχήμα 2.3.2-2: Συστολική Διάταξη για τον Πολλαπλασιασμό Πίνακα με Διάνυσμα (για p=2 και q=3)

Το γενικό σχήμα του παραπάνω σωληνικού αλγόριθμου μπορεί να περιγραφεί ως εξής: Τα y_i, τα οποία αρχικά είναι μηδέν, κινούνται προς τα αριστερά, ενώ τα x_i κινούνται προς τα δεξιά και τα α_{ij} προς τα κάτω. Όλες οι κινήσεις είναι συγχρονισμένες. Κατ' αυτόν τον τρόπο, κάθε y_i μπορεί να συσσωρεύσει όλους τους όρους του, δηλαδή τους όρους $\alpha_{i,i-2}*x_{i-2}$, $\alpha_{i,i-1}*x_{i-1}$, $\alpha_{i,i}*x_i$ και $\alpha_{i,i+1}*x_{i+1}$ πριν να εγκαταλείψει τη διάταξη. Στο σχήμα 2.3.2-3 περιγράφονται τα επτά πρώτα βήματα του αλγόριθμου.



Σχήμα 2.3.2-3: Τα Επτά Πρώτα Βήματα του Αλγόριθμου Πολλαπλασιασμού Πίνακα με Διάνυσμα (για p=2 και q=3)

Είναι προφανές ότι οποιαδήποτε χρονική στιγμή μόνον οι μισοί από τους επεξεργαστές του δικτύου εκτελούν κάποια υπολογιστική δραστηριότητα. Κατά συνέπεια, είναι δυνατόν, συζεύγοντας γειτονικούς στη διάταξη επεξεργαστές να χρησιμοποιηθούν, τελικά, w/2 επεξεργαστές στη διάταξη, για έναν οποιονδήποτε ταινιωτό πίνακα με εύρος ζώνης w.

Στη συνέχεια, ο αλγόριθμος θα περιγραφεί με μεγαλύτερη λεπτομέρεια. Ας υποθέσουμε ότι οι επεξεργαστές είναι αριθμημένοι με τους ακέραιους 1, 2,..., w, αρχίζοντας από τον επεξεργαστή που βρίσκεται στο αριστερό άκρο. Ο κάθε επεξεργαστής διαθέτει τρεις καταχωρητές R_A, R_x, και R_y, οι οποίοι θα αποθηκεύσουν τα στοιχεία του πίνακα A και τα στοιχεία των διανυσμάτων x και y, αντίστοιχα. Αρχικά όλοι οι καταχωρητές περιέχουν μηδενικές τιμές.

Το κάθε βήμα του αλγόριθμου περιλαμβάνει τις παρακάτω λειτουργίες (έχοντας, όμως, υπόψη ότι στα περιττά χρονικά βήματα, μόνο οι αριθμημένοι με περιττό αριθμό επεξεργαστές θα είναι *ενεργοί*, ενώ το αντίστοιχο συμβαίνει για τους αριθμημένους με άρτιο αριθμό επεξεργαστές στα άρτια χρονικά βήματα):

1. *Ολίσθηση* (Shift)

- Ο καταχωρητής R_A δέχεται ένα νέο στοιχείο από τον πίνακα A.
- Ο καταχωρητής R_x δέχεται τα περιεχόμενα του αντίστοιχου καταχωρητή από τον αριστερό γειτονικό του κόμβο. (Ο καταχωρητής R_x, στον επεξεργαστή 1, δέχεται ένα νέο στοιχείο του διανύσματος x).
- Ο καταχωρητής R_y δέχεται τα περιεχόμενα του αντίστοιχου καταχωρητή από το δεξιό γειτονικό του κόμβο. (Ο επεξεργαστής 1 εξάγει τα περιεχόμενα του καταχωρητή του R_y και ο καταχωρητής R_y στον επεξεργαστή w δέχεται μία μηδενική τιμή).

2. *Πολλαπλασιασμός και Πρόσθεση* (Multiply - Add)

- $$R_y \leftarrow R_y + R_A * R_x$$

Χρησιμοποιώντας τον τύπο επεξεργαστή (α) (βλέπε σχήμα 2.3.1.1-1), παρατηρούμε ότι όλες οι λειτουργίες στους επεξεργαστές της διάταξης μπορούν να γίνονται ταυτόχρονα, και ότι, κατά συνέπεια, το κάθε βήμα του αλγόριθμου απαιτεί μία χρονική μονάδα. Με αυτό τον τρόπο μπορεί εύκολα να διαπιστωθεί ότι, μετά από w χρονικές μονάδες, τα στοιχεία του γινομένου $y = Ax$ θα αρχίσουν να εξέρχονται από

τον επεξεργαστή στο αριστερό άκρο της διάταξης, με τη συχνότητα ενός τελικού αποτελέσματος κάθε δύο χρονικές μονάδες.

Κατά συνέπεια, χρησιμοποιώντας τη συγκεκριμένη διάταξη του σχήματος 2.3.2-3, όλα τα στοιχεία του διανύσματος y μπορούν να υπολογιστούν σε 2n+w χρονικές μονάδες, σε σύγκριση με το χρόνο της τάξης του *O(wn)* που απαιτείται για την εκτέλεση του σειριακού αλγόριθμου σε ένα συμβατικό σύστημα von-Neumann.

2.3.3 Αλγόριθμος Πολλαπλασιασμού Πινάκων

Το γινόμενο $C = (c_{ij})$ δύο (nxn) πινάκων $A = (\alpha_{ij})$ και $B = (b_{ij})$ μπορεί να υπολογιστεί από τις παρακάτω επαναληπτικές σχέσεις:

$$\begin{aligned} c_{ij}^{(1)} &= 0 \\ c_{ij}^{(k+1)} &= c_{ij}^{(k)} + \alpha_{ik} * b_{kj} \qquad k = 1,2,...,n \\ c_{ij} &= c_{ij}^{(n+1)} \end{aligned} \tag{2.3.3:1}$$

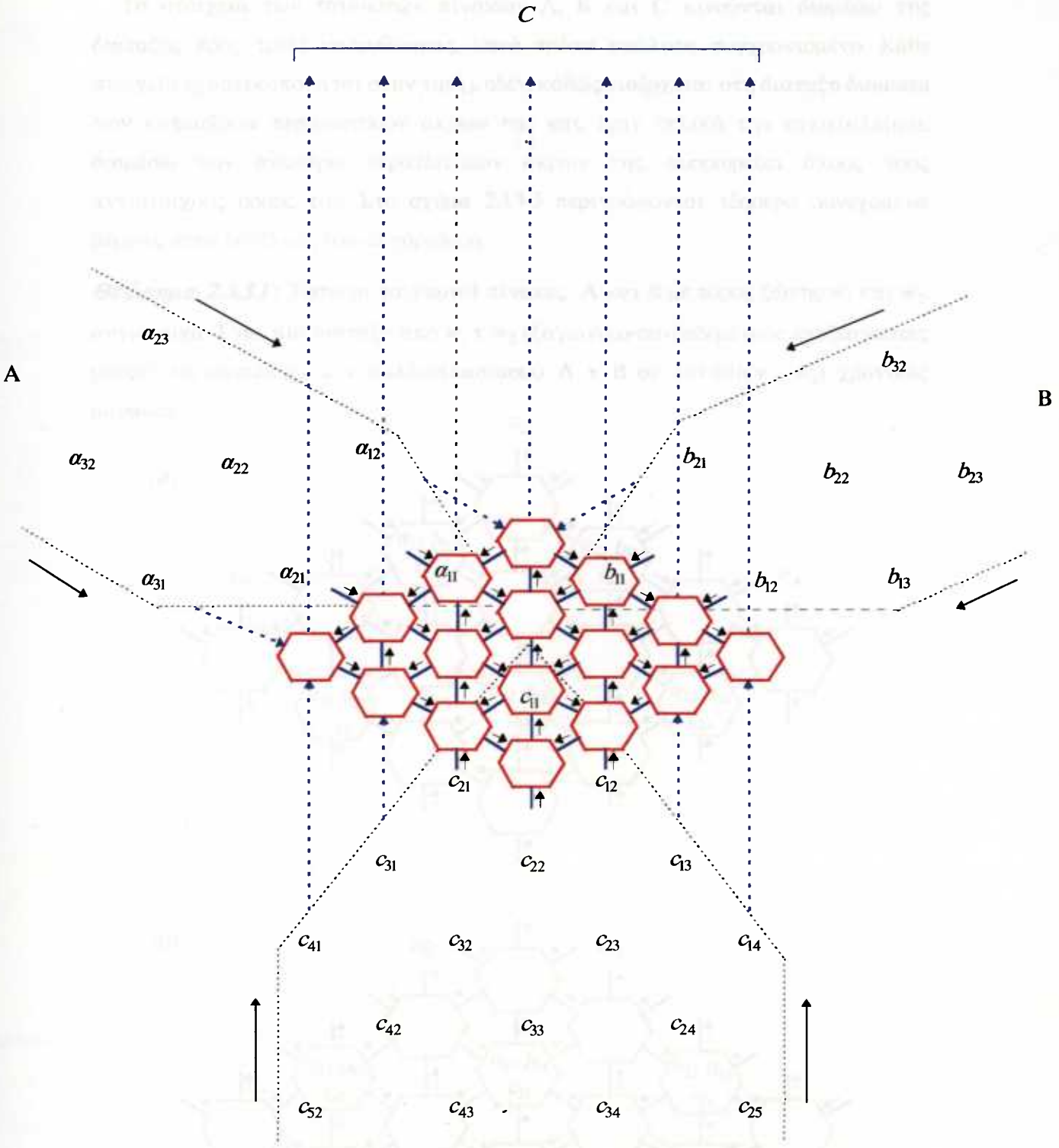
Το εύρος ζώνης των πινάκων A και B θεωρείται ότι ισούται με w_1 και w_2. Στη συνέχεια, θα περιγραφεί διαγραμματικά πως οι παραπάνω επαναληπτικές σχέσεις μπορούν να υπολογιστούν μετατοπίζοντας τα α_{ij}, b_{ij} και c_{ij} διαμέσω ενός πίνακα από εξαγωνικο-συνδεδεμένους επεξεργαστές, χρησιμοποιώντας την τεχνική της σωλήνωσης. Ο αλγόριθμος χρησιμοποιεί τις ίδιες αρχές που περιγράφηκαν στην περίπτωση του αλγόριθμου πολλαπλασιασμού πίνακα με διάνυσμα.

Στα σχήματα 2.3.3-1 και 2.3.3-2 περιγράφονται διαγραμματικά, το πρόβλημα του πολλαπλασιασμού πινάκων και η εξαγωνικο-συνδεδεμένη συστολική διάταξη, αντίστοιχα, όπου η ροή των δεδομένων υποδηλώνεται με βέλη.

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & & & \bigcirc \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & & \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} & \\ & \alpha_{42} & . & & \\ \bigcirc & & & . & \\ & & & & . \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & & \bigcirc \\ b_{21} & b_{22} & b_{23} & b_{24} & \\ & b_{32} & b_{33} & b_{34} & b_{35} \\ & & b_{43} & . & \\ \bigcirc & & & . & \\ & & & & . \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & \bigcirc \\ c_{21} & c_{22} & c_{23} & c_{24} & \\ c_{31} & c_{32} & c_{33} & c_{34} & \\ c_{41} & c_{42} & . & & \\ \bigcirc & & & . & \\ & & & & . \end{bmatrix}$$

A B C

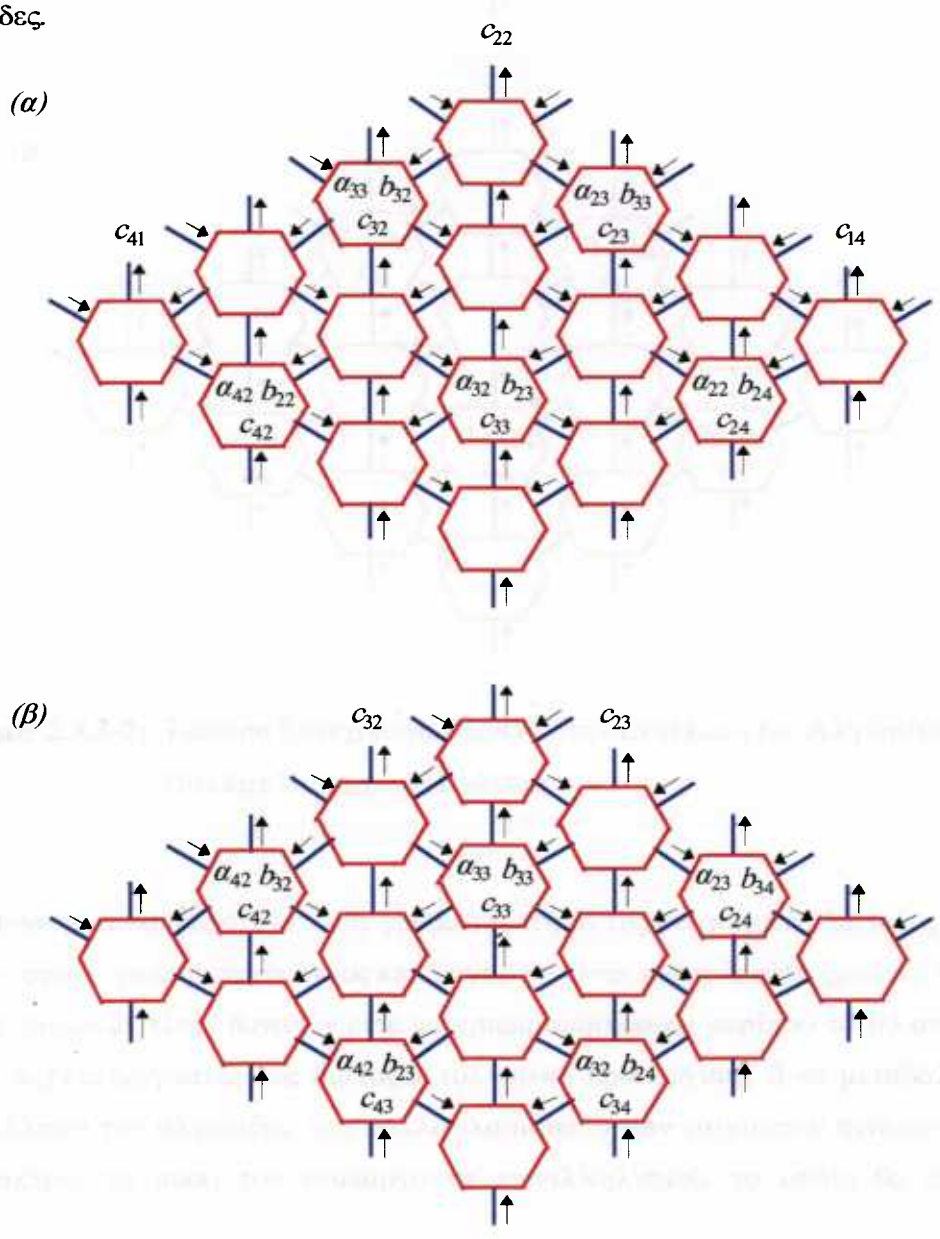
Σχήμα 2.3.3-1: Πολλαπλασιασμός Ταινιωτών Πινάκων

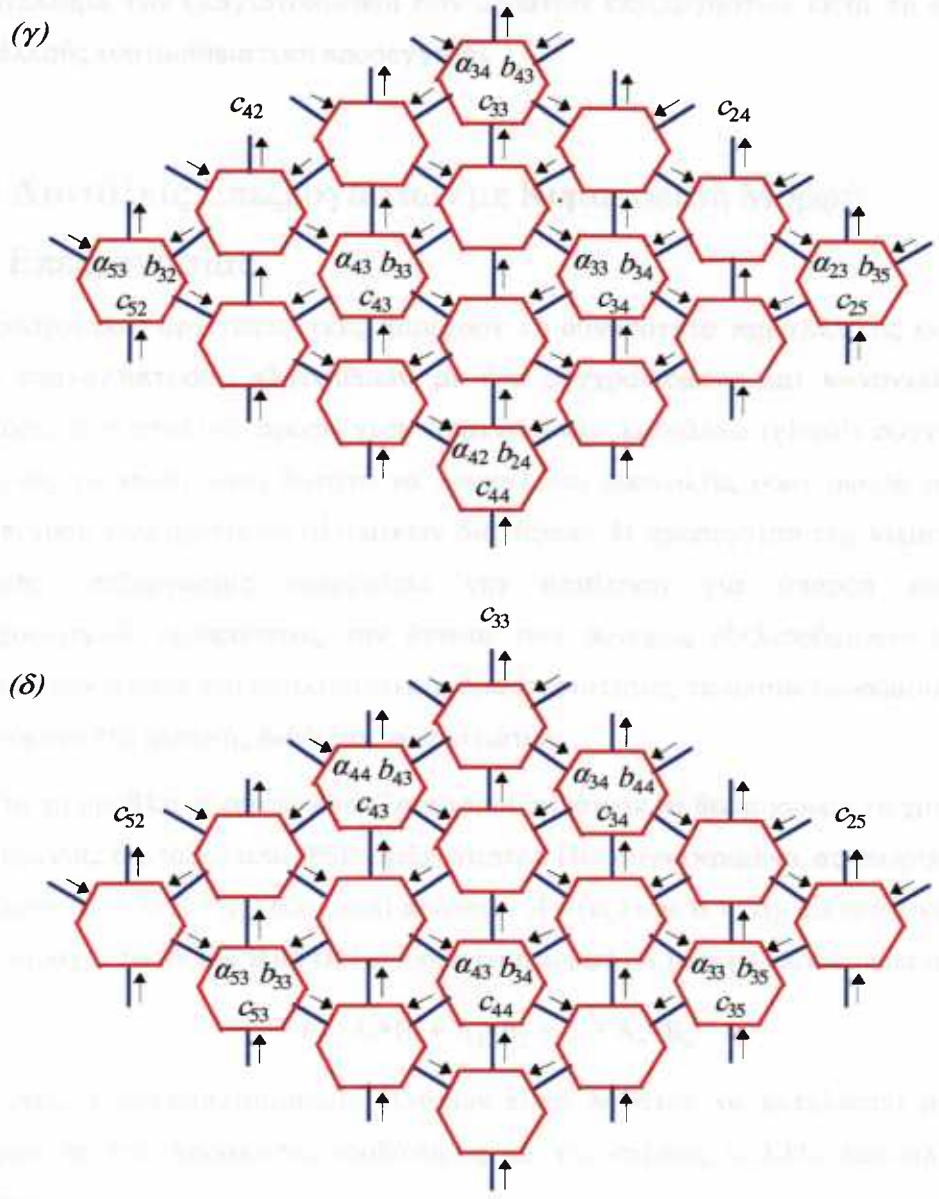


Σχήμα 2.3.3-2: Εξαγωνικο-συνδεδεμένη Συστολική Διάταξη για τον Πολλαπλασιασμό Πινάκων

Τα στοιχεία των ταινιωτών πινάκων A, B και C κινούνται διαμέσω της διάταξης προς τρείς κατευθύνσεις, κατά τρόπο απόλυτα συγχρονισμένο. Κάθε στοιχείο c_{ij} αρχικοποιείται στην τιμή μηδέν, καθώς εισέρχεται στη διάταξη διαμέσω των κατωτέρων περατωτικών άκρων της και, πριν τελικά την εγκαταλείψει, διαμέσω των ανωτέρω περατωτικών άκρων της, συσσωρεύει όλους τους αντίστοιχους όρους του. Στο σχήμα 2.3.3-3 περιγράφονται τέσσερα συνεχόμενα βήματα στην εκτέλεση του αλγόριθμου.

Θεώρημα 2.3.3.1: Έστω οι ταινιωτοί πίνακες A και B με εύρος ζώνης w_1 και w_2, αντίστοιχα. Τότε, μία διάταξη από $w_1 \times w_2$ εξαγωνικο-συνδεδεμένους επεξεργαστές μπορεί να εκτελέσει τον πολλαπλασιασμό A x B σε $3n+\min(w_1, w_2)$ χρονικές μονάδες.





Σχήμα 2.3.3-2: Τέσσερα Συνεχόμενα Βήματα στην Εκτέλεση του Αλγόριθμου Πολλαπλασιασμού Πινάκων

Παρατηρεί κανείς ότι σε κάθε γραμμή ή στήλη της συστολικής διάταξης ο *ένας* μόνον στους *τρείς* συνεχόμενους επεξεργαστές είναι *ενεργός* κάθε χρονική στιγμή. Κατά συνέπεια, είναι δυνατόν είτε να χρησιμοποιήσουμε περίπου το 1/3 από τους (w_1 x w_2) επεξεργαστές της διάταξης (υλισμική προσέγγιση), ή να μεταβάλλουμε κατάλληλα τον αλγόριθμο του πολλαπλασιασμού των ταινιωτών πινάκων, ώστε να αυξηθεί το ποσό του ενυπάρχοντα παραλληλισμού, το οποίο θα έχει ως

αποτέλεσμα την ελαχιστοποίηση των αδρανών επεξεργαστών κατά τη φάση της εκτέλεσής του (μαθηματική προσέγγιση).

2.4 Διατάξεις Επεξεργαστών με Κυματοειδή Μορφή Επεξεργασίας

Οι συστολικές αρχιτεκτονικές παρέχουν τη δυνατότητα παράλληλης εκτέλεσης των επαναληπτικών αλγορίθμων, με ένα συγχρονισμένο και κανονικό τρόπο. Ωστόσο, η συστολική προσέγγιση απαιτεί έναν καθολικό (global) συγχρονισμό, γεγονός το οποίο είναι δυνατό να προκαλέσει δυσκολίες όσον αφορά στη VLSI υλοποίηση των σχετικών υλισμικών διατάξεων. Η προσέγγιση της κυματοειδούς μορφής επεξεργασίας υπερβαίνει την απαίτηση για ύπαρξη καθολικού συγχρονισμού, υιοθετώντας την έννοια των συνεχώς εξελισσόμενων κυμάτων (waves) δεδομένων και υπολογιστικής δραστηριότητας, τα οποία προσομοιάζουν το φαινόμενο της φυσικής διάδοσης των κυμάτων.

Για το πρόβλημα του πολλαπλασιασμού πινάκων θα θεωρήσουμε τη χρήση μιας ορθογώνιας διάταξης από IPSP επεξεργαστές. Πιο συγκεκριμένα, ας θεωρήσουμε το γινόμενο $C = AxB = (c_{ij})$ δύο (nxn) πινάκων $A = (a_{ij})$ και $B = (b_{ij})$. Εκφράζοντας τους δύο παράγοντες A και B με τη διανυσματική μορφή (A_i) και (B_j) προκύπτει ότι:

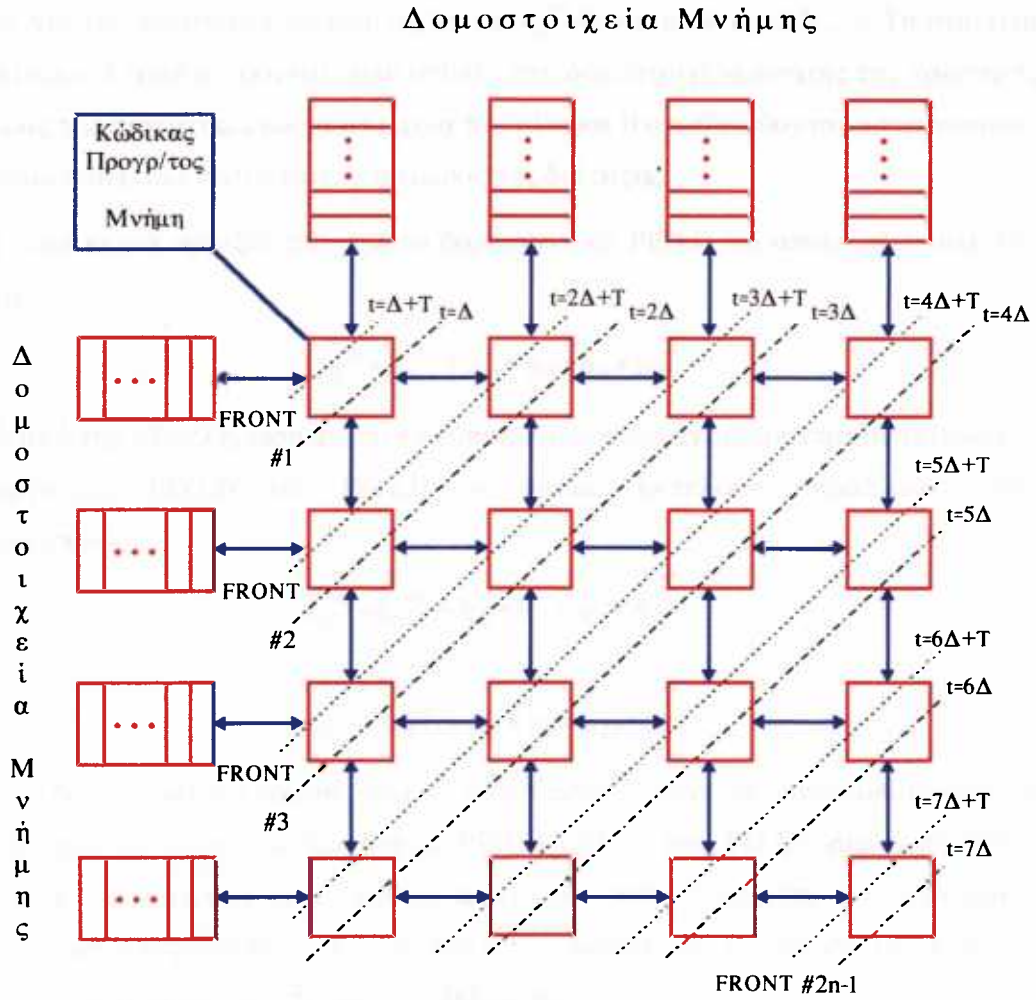
$$C = A_1*B_1 + A_2*B_2 + \ldots + A_n*B_n \qquad (2.4:1)$$

και έτσι, ο πολλαπλασιασμός πινάκων είναι δυνατόν να εκτελεστεί μέσω της εφαρμογής της παρακάτω, ισοδύναμης με τις σχέσεις (2.3.3:1), επαναληπτικής σχέσης:

$$C^{(k)} = C^{(k-1)} + A_k*B_k\,, \qquad (2.4:2)$$

όπου k = 1,2, ..., n.

Από την άποψη της VLSI σχεδίασης, η τοπολογία ενός τέτοιου αλγόριθμου με δεδομένο βαθμό τοπικών διασυνδέσεων και ροή δεδομένων, είναι δυνατόν να απεικονιστεί φυσικά σε έναν τετραγωνικό (nxn) πίνακα, ο οποίος χρησιμοποιεί την έννοια της κυματοειδούς μορφής επεξεργασίας, όπως φαίνεται στο σχήμα 2.4-1.



Σχήμα 2.4-1: Διάρθρωση ενός (nxn) Τετραγωνικού WAP.

Για το συγκεκριμένο αλγόριθμο του πολλαπλασιασμού πινάκων, ισχύει ότι κάθε μαθηματική επανάληψη (recursion) αντιστοιχεί σε μία διαφορετική κυματομορφή στη διάταξη των επεξεργαστών. Κατά συνέπεια, με τη διαδοχική σωλήνωση των κυματομορφών επιτυγχάνεται, τελικά, ο υπολογισμός όλων των βημάτων της επαναληπτικής σχέσης.

Υποθέτοντας ότι όλοι οι καταχωρητές των PEs αρχικοποιούνται στην τιμή μηδέν, για την πρώτη επανάληψη ισχύει ότι $c_{ij}^{(0)}=0$, για κάθε i,j =1, 2, ..., n. Τα στοιχεία του πίνακα Α αποθηκεύονται κατά στήλες στα δομοστοιχεία μνήμης της αριστερής πλευράς της διάταξης, ενώ τα στοιχεία του πίνακα Β αποθηκεύονται κατά γραμμές στα δομοστοιχεία μνήμης στο πάνω μέρος της διάταξης.

Η διαδικασία αρχίζει στο πρώτο βορειοδυτικό PE(1,1), το οποίο υλοποιεί την πράξη

$$c_{11}^{(1)} = c_{11}^{(0)} + \alpha_{11} * b_{11} = \alpha_{11} * b_{11}$$

Μετά από την ολοκλήρωση αυτού του υπολογισμού ενεργοποιούνται οι γειτονικοί επεξεργαστές PE(1,2) και PE(2,1), οι οποίοι εκτελούν παράλληλα τους υπολογισμούς:

$$c_{12}^{(1)} = c_{12}^{(0)} + \alpha_{11} * b_{12} = \alpha_{11} * b_{12},$$

και

$$c_{21}^{(1)} = c_{21}^{(0)} + \alpha_{21} * b_{11} = \alpha_{21} * b_{11}$$

Μετά από την ολοκλήρωση αυτών των υπολογισμών θα ενεργοποιηθούν οι αντίστοιχοι γειτονικοί επεξεργαστές PE(1,3), PE(2,2) και PE(3,1), δημιουργώντας, έτσι, ένα κύμα ιεραρχικών υπολογισμών, το οποίο διασχίζει την ορθογώνια διάταξη των επεξεργαστών. Και σε αυτή την περίπτωση η διάδοση του κύματος προυποθέτει τοπικότητα στη ροή των δεδομένων.

Η πρώτη επανάληψη θεωρείται ότι έχει ολοκληρωθεί, μετά το πρώτο πέρασμα της κυματομορφής από όλα τα κελιά της διάταξης.

Ο ενυπάρχον παραλληλισμός έγκειται στο ότι ταυτόχρονα με την αναδίπλωση του πρώτου κύματος, και αμέσως μετά από την εμφάνιση του πρώτου μέρους του υπολογισμού, μπορεί να αρχίσει η διάδοση του δεύτερου κύματος, δηλαδή, της δεύτερης επανάληψης, και κατόπιν του τρίτου κύματος, κ.ο.κ., μέχρις ότου ολοκληρωθεί ο υπολογισμός του γινομένου των πινάκων Α και Β.

Η σωλήνωση των υπολογισμών είναι εφικτή λόγω του ότι οι κυματομορφές δεν τέμνονται (intersect) ποτέ στο βαθμό που χρησιμοποιούνται διαφορετικοί επεξεργαστές και δεν υφίστανται συγκρούσεις μεταξύ τους. Ουσιαστικά, η αρχιτεκτονική που χρησιμοποιείται αντιστοιχεί στην πλεγματοειδή τοπολογία

των συστημάτων πολλαπλών επεξεργαστών, με τη διαφορά ότι υπάρχουν επιπλέον δομοστοιχεία μνήμης στη δυτική και βόρεια πλευρά της διάταξης.

Τα πλεονεκτήματα αυτής της μορφής επεξεργασίας, έναντι της αντίστοιχης συστολικής, συνοψίζονται στα παρακάτω:

(I) Μειώνεται δραστικά η πολυπλοκότητα στην περιγραφή των παράλληλων αλγορίθμων οι οποίοι αφορούν σε πράξεις μεταξύ πινάκων.

(II) Η σχετική γλώσσα η οποία έχει αναπτυχθεί για τη μηχανή επιτρέπει τη δυνατότητα προγραμματισμού του WAP, διευρύνοντας, έτσι, το φάσμα των εφαρμογών που είναι δυνατόν να υλοποιηθούν.

(III) Η γλώσσα προγραμματισμού αυτή επιτρέπει την προσομοίωση και κατ' επέκταση την επαλήθευση των παράλληλων αλγορίθμων.

(IV) Οι επεξεργαστές διαθέτουν μία δυνατότητα ασύγχρονης αναμονής, η οποία υπακούει στην αρχή του Huyghen ότι οι κυματομορφές δεν τέμνονται.

Κεφάλαιο 3

Σχήματα Συμπύκνωσης Αραιών Πινάκων

3.1 Εισαγωγή

Ένας *αραιός πίνακας* (sparse matrix) είναι ένας πίνακας του οποίου τα στοιχεία στο μεγαλύτερο ποσοστό τους είναι ίσα με το μηδέν. Ένα από τα πλέον κρίσιμα προβλήματα, που κατά τις τελευταίες δεκαετίες έχει απασχολήσει την επιστημονική κοινότητα, είναι ο χειρισμός αραιών πινάκων μεγάλου μεγέθους κατά τρόπον ώστε να επιτυγχάνεται σημαντική μείωση σε ότι αφορά στις απαιτήσεις αποθήκευσής τους, αλλά και στο απαιτούμενο πλήθος πράξεων που συνεπάγεται η χρήση τους κατά την επίλυση κάποιου προβλήματος. Έχει αποδειχθεί ότι η λύση πολλών προβλημάτων ανάγεται, τελικά, στην επίλυση ενός γραμμικού συστήματος της μορφής:

$$A\,x = y\,, \qquad (3.1{:}1)$$

όπου ο συντελεστής πίνακας A είναι ένας (nxn) αραιός, θετικά ορισμένος συμμετρικός πίνακας. Σε τέτοιου είδους γραμμικά συστήματα, αποφασιστικό ρόλο παίζει η δημιουργία ενός *πίνακα μετάθεσης* (permutation matrix) P, ο οποίος να περιορίζει το ποσοστό των μη μηδενικών στοιχείων (fill-ins) που εμφανίζονται στον πίνακα A κατά τη διαδικασία της παραγοντοποίησής του. Εξαιτίας του ότι ο

πίνακας PAP^T είναι επίσης συμμετρικός και θετικά ορισμένος για οποιονδήποτε πίνακα μετάθεσης P, το παρακάτω σύστημα είναι δυνατό να λυθεί εναλλακτικά

$$(PAP^T)(Px) = (Pb) \qquad (3.1:2)$$

Από την εκάστοτε επιλογή του πίνακα P εξαρτάται τόσο το ποσοστό των μη μηδενικών στοιχείων που θα εμφανιστούν κατά τη διαδικασία της παραγοντοποίησης του πίνακα PAP^T, όσο και το *εύρος ζώνης* (bandwidth) του πίνακα αυτού.

Το εύρος ζώνης ενός πίνακα A ορίζεται ως εξής:

$$\beta = \max\{ r_i(A) : i=1,...,n \} \qquad (3.1:3)$$

όπου το $r_i(A) = i - f_i(A)$ ορίζει το εύρος της ι-οστής γραμμής του A, το $f_i(A) = \min\{ j : j \in row(i) \}$ ορίζει το αριστερότερο μη μηδενικό στοιχείο της ι-οστής γραμμής του A και το $row(i) = \{ j : a_{ij} \neq 0, \text{ and } 1 \leq j \leq i \}$. Αντίστοιχα, η *κατατομή* (profile) ενός πίνακα A ορίζεται ως εξής :

$$\sum r_i, \quad \text{for } i=1,...,n \qquad (3.1:4)$$

Ένας αριθμός προσεγγίσεων έχουν προταθεί και υιοθετηθεί για τη συμπύκνωση, δηλαδή, τη μείωση του εύρους ζώνης και της κατατομής αραιών πινάκων γενικής μορφής. Οι προσεγγίσεις αυτές, οι οποίες θα παρουσιαστούν συνοπτικά στις επόμενες Παραγράφους αυτού του Κεφαλαίου, κατά κύριο λόγο αφορούν σε σειριακούς αλγόριθμους οι οποίοι επιδιώκουν την εύρεση ενός κατάλληλου πίνακα μετάθεσης P που θα χρησιμοποιηθεί για τη δημιουργία ενός πίνακα PAP^T με μικρότερο εύρος ζώνης και κατατομή σε σχέση με τον αρχικό αραιό πίνακα A. Στη συνέχεια, δίνεται μία συνοπτική περιγραφή των προσεγγίσεων και των αλγορίθμων οι οποίοι θα αναλυθούν εκτενέστερα και σε συγκριτική βάση στις επόμενες Παραγράφους αυτού του Κεφαλαίου.

Ο επαναληπτικός αλγόριθμος των Barnard, et al, [3] εκμεταλλεύεται συγκεκριμένες ιδιότητες του συνόλου των ιδιοτιμών (eigenvalues) του αντίστοιχου πίνακα Laplace, υπολογίζοντας ένα *διάνυσμα μετάθεσης* το οποίο είναι όσο το δυνατόν πιο κοντά στο δεύτερης τάξης ιδιοδιάνυσμα (eigenvector) Laplace. Το βασικότερο πλεονέκτημα αυτής της προσέγγισης είναι ότι χρησιμοποιεί τυπικές πράξεις κινητής υποδιαστολής, γεγονός που επιτρέπει την άμεση εκμετάλλευση του ενυπάρχοντος παραλληλισμού στον αλγόριθμο. Η μείωση του εύρους ζώνης που επιτυγχάνεται εξαρτάται άμεσα από το πλήθος των επαναλήψεων που εκτελεί

ο αλγόριθμος. Κατά συνέπεια, η καλύτερη δυνατή μείωση του εύρους ζώνης εξαρτάται από την κατάλληλη επιλογή του κριτηρίου τερματισμού της επαναληπτικής διαδικασίας.

Ο αλγόριθμος του Rosen, [63] χρησιμοποιεί μία τεχνική μετάθεσης γραμμών/στηλών για τη μείωση του εύρους ζώνης αραιών πινάκων, ενώ ο αλγόριθμος των Akhras, et al, [1] εξετάζει συγκεκριμένα χαρακτηριστικά του *πίνακα συνεκτικότητας κορυφών* (node connectivity matrix) τα οποία και χρησιμοποιεί ως κριτήρια για την *επαναδιάταξη* (re-ordering) των γραμμών και των αντίστοιχων στηλών του αρχικού αραιού πίνακα.

Τέλος, παρουσιάζεται η εντελώς διαφορετική αλγοριθμική προσέγγιση του Megson, [56], η οποία αναφέρεται σε αραιούς πίνακες συγκεκριμένου τύπου που προκύπτουν από παραβολικές και ελλειπτικές εξισώσεις δευτέρου και τρίτου βαθμού. Στην προσέγγιση αυτή η συμπύκνωση του εκάστοτε συντελεστή πίνακα οδηγεί σε μία φυσική συμπύκνωση της αντίστοιχης εξαγωνικο-συνδεδεμένης (hex-connected) συστολικής αρχιτεκτονικής που χρησιμοποιείται για την παραγοντοποίησή του.

3.2 Επαναληπτικά Σχήματα Συμπύκνωσης

Ο πίνακας γειτνίασης ενός γραφήματος, ορίζεται ως ένας (nxn) πίνακας στον οποίο κάθε στοιχείο του (i,j) ισούται με το πλήθος των *ακμών* (edges) που συνδέουν τις *κορυφές* (vertices) i και j. Ένας αραιός πίνακας μπορεί να μετατραπεί, άμεσα, σε έναν πίνακα γειτνίασης αν αντικαταστήσουμε όλα τα μη μηδενικά του στοιχεία με μονάδες. Είναι προφανές ότι η κύρια διαγώνιος του πίνακα γειτνίασης αποτελείται από μηδενικά στοιχεία εφόσον το αντίστοιχο γράφημα δεν περιέχει *βρόχους* (loops). Οι αλγόριθμοι συμπύκνωσης αραιών πινάκων χρησιμοποιούν ως *είσοδο* (input) έναν πίνακα γειτνίασης ελαφρά τροποποιημένο από την άποψη ότι η κύρια διαγώνιός του δεν είναι μηδενική. Οι αλγόριθμοι αυτοί ασχολούνται με την επαναδιάταξη των μη μηδενικών στοιχείων του υπό εξέταση συντελεστή πίνακα· δηλαδή την επαναδιάταξη των μονάδων του αντίστοιχου πίνακα γειτνίασης, χωρίς να εξετάζουν τις πραγματικές τιμές του συντελεστή πίνακα οι οποίες αντιστοιχούν σε αυτές τις μονάδες.

3.2.1 Φασματικός (Spectral) Αλγόριθμος Μείωσης του Φακέλλου (Envelope)

Ο συγκεκριμένος αλγόριθμος, ο οποίος προτάθηκε σχετικά πρόσφατα, επιτυγχάνει τη μείωση του εύρους ζώνης συμμετρικών αραιών πινάκων μεγάλου μεγέθους, κάνοντας χρήση των ιδιοτήτων του *φάσματος* (spectrum), που ορίζεται ως το σύνολο των ιδιοτιμών του αντίστοιχου πίνακα Laplace. Δοθέντος ενός μη κατευθυνόμενου γραφήματος G=(V,E), ο πίνακας Laplace, L, ορίζεται ως ένας (nxn) πίνακας ο οποίος προκύπτει από τη διαφορά των πινάκων D και B. Ο πίνακας D είναι ένας διαγώνιος πίνακας στον οποίο τα στοιχεία της διαγωνίου αντιστοιχούν στους βαθμούς των κορυφών του γραφήματος, ενώ ο πίνακας B είναι ο πίνακας γειτνίασης του γραφήματος. Έτσι, ο πίνακας Laplace, L=D-B, ορίζεται με τον ακόλουθο τρόπο:

$$l_{ij} = \begin{cases} -1, \text{ if } a_{ij} \neq 0 \\ d_v(G), \text{ if } i=j \\ 0, \text{ otherwise} \end{cases}$$

Οι ιδιοτιμές του πίνακα L αποτελούν τις ιδιοτιμές Laplace του γραφήματος G, $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Το ιδιοδιάνυσμα το οποίο αντιστοιχεί στην ιδιοτιμή λ_k συμβολίζεται με x_k και ονομάζεται το k-οστό ιδιοδιάνυσμα του πίνακα L. Εξαιτίας του ότι ο πίνακας L είναι *ιδιότυπος* (singular), οι ιδιοτιμές του είναι μη αρνητικές, ενώ για τη μικρότερη ιδιοτιμή ισχύει ότι $\lambda_1=0$ και έτσι το αντίστοιχο ιδιοδιάνυσμα μπορεί να είναι οποιοδήποτε σταθερό και μη μηδενικό ιδιοδιάνυσμα c. Αν το γράφημα G είναι συνεκτικό, τότε ο πίνακας L είναι μη περαιτέρω απλουστεύσιμος (irreducible) και ισχύει ότι $\lambda_2>0$. Έχει αποδειχθεί ότι οι μικρότερες μη μηδενικές ιδιοτιμές και τα αντίστοιχα ιδιοδιανύσματά τους έχουν πολύ σημαντικές ιδιότητες οι οποίες τα καθιστούν εξαιρετικά χρήσιμα για τη λύση διαφόρων προβλημάτων.

Δύο παράμετροι οι οποίες σχετίζονται με το εύρος ζώνης ενός πίνακα A, είναι το *μέγεθος φακέλλου* (*envelope size*) και ο εργασιακός φάκελλος (*envelope work*). Η πρώτη παράμετρος ορίζεται ως το σύνολο των δεικτών οι οποίοι βρίσκονται μεταξύ του δείκτη της στήλης που αντιστοιχεί στο αριστερότερο μη μηδενικό στοιχείο και του αντίστοιχου δείκτη που αντιστοιχεί στο διαγώνιο στοιχείο κάθε γραμμής, δηλαδή,

$$Env(A) = \{ (i,j) : f_i(A) \le j \le i, i=1,...,n \} \quad (3.2.1:1)$$

Το *μέγεθος* αυτής της παραμέτρου συμβολίζεται με Esize(A) = |Env(A)|. Η δεύτερη παράμετρος, ο εργασιακός φάκελλος, ορίζεται από τη σχέση:

$$Ework(A) = \sum_{i=1}^{n} r_i^2 \quad (3.2.1:2)$$

Η παράμετρος αυτή, συνήθως, χρησιμοποιείται ως ένα άνω φράγμα του υπολογιστικού έργου που απαιτεί η διαδικασία παραγοντοποίησης του πίνακα A με βάση τον αλγόριθμο Cholesky, όταν χρησιμοποιείται το συγκεκριμένο σχήμα αποθήκευσης. Θα πρέπει να σημειωθεί ότι όλες αυτές οι παράμετροι εξαρτώνται αυστηρά από την αρχική διάταξη των γραμμών και των στηλών του πίνακα A και γι' αυτό είναι δυνατόν οι τιμές τους να διαφέρουν σε ένα συμμετρικό πίνακα PAP^T, όπου το P είναι ένας πίνακας μετάθεσης.

Οι παράμετροι μέγεθος φακέλου και εργασιακός φάκελλος σχετίζονται, επίσης, με τις ποσότητες 1-sum, $\sigma_1(A)$ and 2-sum, $\sigma_2(A)^2$, ως εξής:

$$Esize(A) = \sum_{i=1}^{n} \max_{j \in row(i)} (i-j) \quad (3.2.1:3)$$

$$\sigma_1(A) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i-j) \quad (3.2.1:4)$$

$$Ework(A) = \sum_{i=1}^{n} \max_{j \in row(i)} (i-j)^2 \quad (3.2.1:5)$$

$$\sigma_2^2(A) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i-j)^2 \quad (3.2.1:6)$$

Η μείωση του φακέλλου ενός πίνακα συνεπάγεται και την ελαχιστοποίηση της παραμέτρου Ework(A) και κατά συνέπεια την ελαχιστοποίηση του υπολογιστικού έργου που απαιτείται κατά την εφαρμογή της διαδικασίας παραγοντοποίησης Cholesky.

Έστω $\mathcal{P}$ ένα σύνολο από n-διανύσματα $\underline{p}$ τα στοιχεία των οποίων αποτελούν μεταθέσεις του συνόλου $\{-(n-1)/2,...,-1,0,1,...,(n-1)/2\}$, όταν το n είναι περιττός αριθμός, ή του συνόλου $\{-n/2,...,-1,+1,...,n/2\}$, όταν το n είναι άρτιος. Κάνοντας χρήση των διανυσμάτων του συνόλου $\mathcal{P}$, το πρόβλημα 2-sum ενός συμμετρικού πίνακα A ορίζεται με βάση την ακόλουθη σχέση:

$$\min_{\underline{x}\in\mathcal{P}} = \sum_{i=1}^{n} \sum_{j\in row(i)} (x_i - x_j)^2 = 1/2 \min_{\underline{x}\in\mathcal{P}} \sum_{a_{ij}\neq 0} (x_i - x_j)^2 \qquad (3.2.1:7)$$

Προκειμένου να προσεγγιστεί αυτό το δύσκολο διακριτό πρόβλημα, θα πρέπει να διευρυνθεί η συνθήκη $\underline{x}\in\mathcal{P}$, έτσι ώστε η ελαχιστοποίηση της αντικειμενικής συνάρτησης να γίνεται σε σχέση με ένα καταλληλότερο σύνολο n-διανυσμάτων.

Για κάθε $\underline{p}\in\mathcal{P}$, ισχύει ότι $\underline{p}^T\underline{u}=0$, και $l \equiv \underline{p}^T\underline{p} = (n/12)(n^2-1)$, για περιττό n καθώς και $l \equiv \underline{p}^T\underline{p} = (n/12)(n+1)(n+2)$, για άρτιο n, όπου $\underline{u}=(1,1,...,1)^T$. Δοθέντος ενός διανύσματος $\underline{x}\in\Re^n$, ο ορισμός ενός διανύσματος μετάθεσης (permutation vector) $\underline{p}$, το οποίο προκύπτει από το $\underline{x}$, μπορεί να γίνει με βάση τον κανόνα : $p_i \leq p_j$ *αν και μόνο αν,* $x_i \leq x_j$ Κατά συνέπεια, μία συνεχής διεύρυνση του διακριτού προβλήματος μπορεί να προκύψει θεωρώντας το σύνολο X των διανυσμάτων $\underline{x}\in\Re^n$ τα οποία ικανοποιούν τις σχέσεις $\underline{x}\neq\underline{0}$, $\underline{x}^T\underline{u}=0$ and $\underline{x}^T\underline{x}=l$. Έτσι, το πρόβλημα συνεχούς βελτιστοποίησης (continuous optimization problem) που προκύπτει είναι το εξής:

$$1/2 \min_{\underline{x}\in\mathcal{X}} \sum_{a_{ij}\neq 0} (x_i - x_j)^2 = \min_{\underline{x}\in\mathcal{X}} \left[\sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{\substack{j<i \\ a_{ij}\neq 0}} x_i x_j \right]$$

$$= \min_{\underline{x}\in\mathcal{X}} \underline{x}^T D \underline{x} - \underline{x}^T B \underline{x} = \min_{\underline{x}\in\mathcal{X}} \underline{x}^T Q \underline{x}$$

$$= \lambda_2 \underline{x}_2^T \underline{x}_2 = \lambda_2 l$$

Συνεπώς, το ιδιοδιάνυσμα Laplace $\underline{x}_2$, δεύτερης τάξης, επιτυγχάνει μία συνεχή προσέγγιση του προβλήματος 2-sum.

Ο φασματικός αλγόριθμος εφαρμόζει και αυτός μία τεχνική επαναδιάταξης των κορυφών του αντίστοιχου γραφήματος γειτνίασης (adjacency graph) του αρχικού αραιού πίνακα A. Ο συγκεκριμένος αλγόριθμος υπολογίζει ένα διάνυσμα μετάθεσης το οποίο είναι όσο το δυνατόν πιο κοντά, από την άποψη της δεύτερης νόρμας και σε σχέση με οποιοδήποτε άλλο τέτοιο διάνυσμα, στο δεύτερης τάξης ιδιοδιάνυσμα Laplace. Κατά συνέπεια, το παραγόμενο ιδιοδιάνυσμα μπορεί να θεωρηθεί σαν μία προσεγγιστική λύση του συνδυαστικού προβλήματος.

Οι διακεκριμένες φάσεις του φασματικού αλγόριθμου είναι οι εξής:

(I) Με βάση την αραιή δομή του πίνακα A, διαμόρφωσε τον πίνακα L.

(II) Υπολόγισε το δεύτερης τάξης ιδιοδιάνυσμα x_2 του πίνακα L.

(III) Ταξινόμησε τα στοιχεία του ιδιοδιανύσματος κατά μη φθίνουσα (non-decreasing) σειρά και επαναδιάταξε τον πίνακα A χρησιμοποιώντας το αντίστοιχο διάνυσμα μετάθεσης. Επίσης, ταξινόμησε τα στοιχεία του ιδιοδιανύσματος κατά μη αύξουσα (non-increasing) σειρά και επαναδιάταξε, κατά τον ίδιο τρόπο, τον πίνακα A. Επέλεξε εκείνη τη μετάθεση του διανύσματος η οποία παράγει το μικρότερο μέγεθος φακέλλου.

Θα πρεπει να σημειωθεί ότι, οι ελάχιστες τιμές για το εύρος ζώνης, το μέγεθος φακέλλου και τον εργασιακό φάκελλο δεν επιτυγχάνονται, ταυτόχρονα πάντοτε μέσω της χρήσης του ίδιου πίνακα μετάθεσης P. Ωστόσο, η μείωση της παραμέτρου μέγεθος φακέλλου συνεπάγεται, συνήθως, μία αποτελεσματική μείωση του εύρους ζώνης του αρχικού αραιού πίνακα. Η επαναληπτική φύση του αλγόριθμου επιτρέπει, μέσω της κατάλληλης επιλογής του κριτηρίου τερματισμού της επαναληπτικής διαδικασίας, την επίτευξη καλύτερων αποτελεσμάτων όσον αφορά στη μείωση των τιμών των παραμέτρων που αναφέρθηκαν και κατ' επέκταση στη μείωση των απαιτήσεων αποθήκευσης, σε βάρος, βέβαια, του απαιτούμενου χρόνου, και αντίστροφα. Από την άλλη μεριά, όμως, ο αλγόριθμος αυτός απαιτεί σημαντικά μεγαλύτερο χρόνο εκτέλεσης για τον υπολογισμό της κατάλληλης διατάξης των γραμμών του αραιού πίνακα, σε σχέση με το χρόνο που απαιτούν οι αλγόριθμοι των Cuthill-Mckee και Gibbs, et al, οι οποίοι θα συζητηθούν σε επόμενες Παραγράφους.

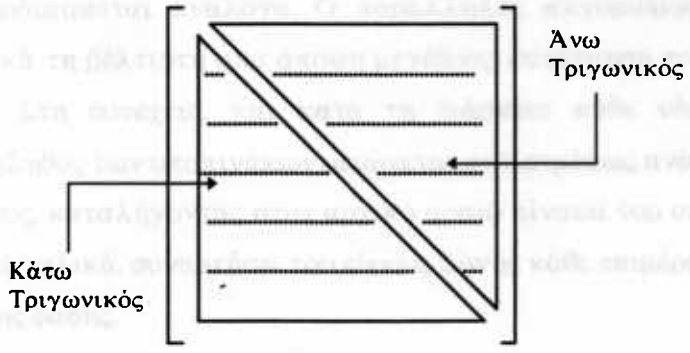
3.3 Μη Επαναληπτικά Σχήματα Συμπύκνωσης

3.3.1 Αλγόριθμοι Αντιμετάθεσης Γραμμών/Στηλών

Στον αλγόριθμο του Rosen, ο βασικός στόχος είναι η επίτευξη μιας κατάλληλης ακολουθίας αντιμεταθέσεων γραμμών/στηλών του αραιού πίνακα A, έτσι ώστε τα στοιχεία του να συγκεντρωθούν γύρω από την κύρια διαγώνιο. Χρησιμοποιώντας ως κριτήριο την τιμή του εύρους ζώνης, προσδιορίζεται κάθε φορά το κατάλληλο ζεύγος (γραμμής/στήλης) που θα πρέπει να αντιμετατεθεί προκειμένου η τιμή αυτή να γίνει η ελάχιστη δυνατή. Ο αλγόριθμος είναι σχετικά απλός και υποδιαιρείται σε δύο κύριες φάσεις, οι οποίες εκτελούνται

επαναληπτικά μέχρις ότου να ισχύσουν κάποιες συνθήκες. Κατά τη διάρκεια της πρώτης φάσης, προσδιορίζονται και αντίστοιχα εκτελούνται όλες οι πιθανές αντιμεταθέσεις μεταξύ γραμμών/στηλών οι οποίες μειώνουν το αρχικό εύρος ζώνης. Η δεύτερη φάση αρχίζει όταν πλέον δεν υπάρχουν άλλες τέτοιου είδους αντιμεταθέσεις που να μειώνουν το εύρος ζώνης του αρχικού αραιού πίνακα. Ο σκοπός της δεύτερης αυτής φάσης είναι να εντοπιστούν και οι αντιμεταθέσεις εκείνες των γραμμών/στηλών οι οποίες δε μειώνουν περαιτέρω, αλλά διατηρούν σταθερό το εύρος ζώνης του αραιού πίνακα. Με αυτόν τον τρόπο, επιδιώκεται μία πιθανώς μεγαλύτερη μείωση του εύρους ζώνης μέσω της εκ νέου εφαρμογής της πρώτης φάσης του αλγόριθμου. Το κριτήριο τερματισμού της δεύτερης φάσης δεν επιτρέπει την αντιμετάθεση δύο γραμμών/στηλών που έχουν ήδη αντιμετατεθεί.

Η συμμετρική μορφή των αραιών πινάκων που εξετάζονται επιτρέπει την εφαρμογή του αλγόριθμου αυτού στο άνω ή κάτω μόνο τριγωνικό τμήμα τους (βλέπε σχήμα 3.3.1-1). Το τμήμα του πίνακα που, τελικά, επιλέγεται ως είσοδος στον αλγόριθμο απεικονίζεται σε έναν *πίνακα θέσεων*, στον οποίο δηλώνονται οι θέσεις όλων των μη μηδενικών στοιχείων του τμήματος αυτού. Τα στοιχεία αυτού του πίνακα θέσεων είναι τα δεδομένα εισόδου του αλγόριθμου και τροποποιούνται κάθε φορά ανάλογα με τη διαδικασία επαναδιάταξης των γραμμών/στηλών η οποία λαμβάνει χώρα. Μετά την ολοκλήρωση του αλγόριθμου, τα στοιχεία του πίνακα θέσεων περιέχουν τις τελικές θέσεις των στοιχείων του αντίστοιχου άνω ή κάτω τριγωνικού τμήματος του αραιού πίνακα που αρχικά επιλέχθηκε. Με την εφαρμογή της αντίστροφης διαδικασίας τα περιεχόμενα του πίνακα θέσεων αποκωδικοποιούνται και προκύπτει σε συμπυκνωμένη μορφή ο αρχικά αραιός πίνακας.



Σχήμα 3.3.1-1: Άνω και Κάτω Τριγωνικό Τμήμα ενός Πίνακα

Η αλγοριθμική διαδικασία περιγράφεται συνοπτικά από τα ακόλουθα βήματα :

(I) Προσδιόρισε το μέγιστο εύρος ζώνης και το ζεύγος(γραμμή, στήλη) που το παράγει.

(II) Έλεγξε αν είναι δυνατή η πραγματοποίηση μιας αντιμετάθεσης μεταξύ του ζεύγους(γραμμή, στήλη) που παράγει το μέγιστο εύρος ζώνης και οποιουδήποτε άλλου ζεύγους, έτσι ώστε το εύρος ζώνης να μειώνεται.

(III) Αν υπάρχει μία τέτοια αντιμετάθεση, τότε πραγματοποίησέ την και αποθήκευσε τις τροποποιημένες θέσεις των στοιχείων που προέκυψαν στον πίνακα θέσεων. Κατόπιν, επίστρεψε στο βήμα (I), διαφορετικά συνέχισε με το βήμα (IV).

(IV) Έλεγξε αν είναι δυνατή η πραγματοποίηση μιας αντιμετάθεσης μεταξύ του ζεύγους(γραμμή, στήλη) που παράγει το μέγιστο εύρος ζώνης και οποιουδήποτε άλλου ζεύγους, έτσι ώστε το εύρος ζώνης να διατηρείται σταθερό.

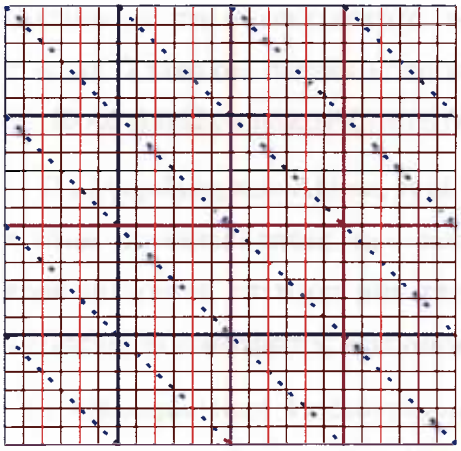
(V) Αν υπάρχει μία τέτοια αντιμετάθεση, τότε πραγματοποίησέ την και αποθήκευσε τις τροποποιημένες θέσεις των στοιχείων που προέκυψαν στον πίνακα θέσεων. Κατόπιν επίστρεψε στο βήμα (I), διαφορετικά τερμάτισε τον αλγόριθμο.

Μία βελτιωμένη παράλληλη προσέγγιση του αλγόριθμου του Rosen, η οποία έχει προταθεί από τους (Efremides, et al, [33]), θεωρεί μία κατάλληλη κατάτμηση του αρχικού αραιού πίνακα και κατόπιν εφαρμόζει τον αλγόριθμο σε κάθε ξεχωριστό *υποπίνακα* που προκύπτει. Η επεξεργασία των υποπινάκων εκτελείται παράλληλα και τα μη μηδενικά στοιχεία του καθενός συγκεντρώνονται γύρω από την κύρια διαγώνιό του. Σε κάθε νέα επανάληψη της όλης διαδικασίας το πλήθος των υποπινάκων υποδιαιρείται ανάλογα. Ο παράλληλος αλγόριθμος εφαρμόζεται θεωρώντας αρχικά τη βέλτιστη από άποψη μεγέθους κατάτμηση του υπό εξέταση αραιού πίνακα. Στη συνέχεια, και κατά τη διάρκεια κάθε νέας φάσης του αλγόριθμου, το πλήθος των υποπινάκων μειώνεται αντιστρόφως ανάλογα σε σχέση με το μέγεθός τους, καταλήγοντας στον αρχικό αραιό πίνακα του οποίου το εύρος ζώνης θα μειωθεί, τελικά, συναρτήσει του εύρους ζώνης κάθε επιμέρους υποπίνακα της προηγούμενης φάσης.

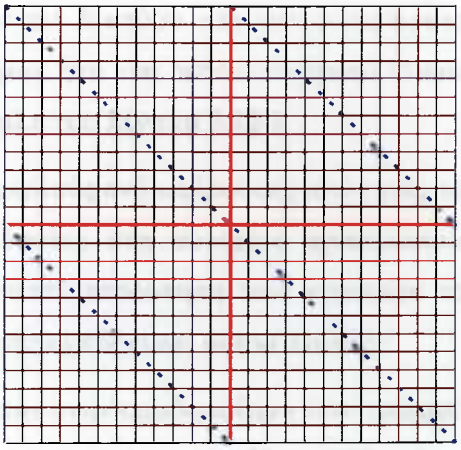
Ένα επιπλέον βασικό πλεονέκτημα αυτής της προσέγγισης είναι ότι μπορεί να εφαρμοστεί και σε μη συμμετρικούς αραιούς πίνακες, με τη διαφορά ότι σε αυτή την

περίπτωση ο αλγόριθμος εφαρμόζεται τόσο στο άνω όσο και στο κάτω τριγωνικό τμήμα κάθε υποπίνακα. Κατόπιν ως εύρος ζώνης του τελικού συμπυκνωμένου πίνακα λαμβάνεται το μέγιστο εύρος ζώνης που προκύπτει είτε από το άνω ή από το κάτω τριγωνικό τμήμα του εκάστοτε υποπίνακα. Με τον τρόπο αυτό, επιτυγχάνεται μία σημαντική, αν και όχι πάντοτε βέλτιστη, μείωση του εύρους ζώνης του αρχικού αραιού πίνακα.

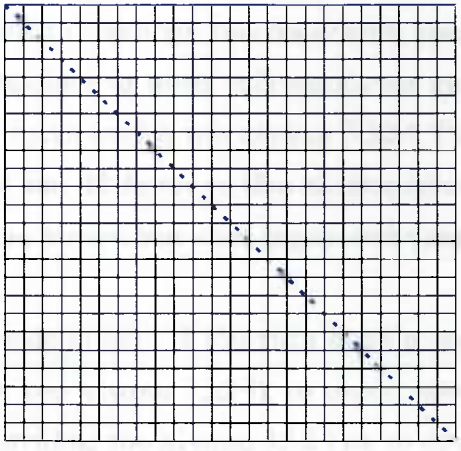
Στο σχήμα 3.3.1-2 παρουσιάζεται μία πιθανή κατάτμηση ενός γενικής μορφής αραιού πίνακα μεγέθους (24x24), καθώς επίσης και τα αντίστοιχα άνω και κάτω τριγωνικά τμήματα για κάθε υποπίνακα που προκύπτει.



Βήμα 1



Βήμα 2



Βήμα 3 (τελικό)

Σχήμα 3.3.1-2: Κατάτμηση Αρχικού Αραιού Πίνακα

3.3.2 Αλγόριθμος Επανεττικετοδότησης (Relabelling) Κορυφών

Όπως έχει ήδη αναφερθεί, αποφασιστικό ρόλο στο πρόβλημα της ελαχιστοποίησης του εύρους ζώνης ενός αραιού πίνακα ή ενός δικτύου επικοινωνίας (communication network) παίζει η έννοια του πίνακα γειτνίασης (adjacency matrix), ο οποίος ως γνωστόν είναι ένας πίνακας του οποίου τα στοιχεία είναι 1 ή 0 και υποδηλώνουν τη συνεκτικότητα και μη συνεκτικότητα των κορυφών, αντίστοιχα. Ο πίνακας αυτός, αν και μετασχηματίζεται με ποικίλλους τρόπους κάθε φορά, περιέχει πάντοτε τα δεδομένα *είσοδου* των αλγόριθμων ελαχιστοποίησης του εύρους ζώνης. Έτσι, μερικοί αλγόριθμοι μετασχηματίζουν τον πίνακα γειτνίασης σε έναν πίνακα συνεκτικότητας των κορυφών που περιέχει όλες τις απαιτούμενες πληροφορίες σχετικά με τη συνεκτικότητα του θεωρούμενου δικτύου επικοινωνίας/αραιού πίνακα. Ο πίνακας συνεκτικότητας των κορυφών, για ένα δεδομένο αραιό πίνακα μεγέθους (12x12), απεικονίζεται στη δεύτερη στήλη του πίνακα 3.3.2-1.

Ο αλγόριθμος των Akhras, et al, [1] εξετάζει μερικές ενδιαφέρουσες ιδιότητες του πίνακα συνεκτικότητας των κορυφών, τις οποίες στη συνέχεια χρησιμοποιεί προκειμένου να επιτύχει μία αποδοτική επανεττικετοδότηση των κορυφών του αντίστοιχου γραφήματος γειτνίασης. Οι ιδιότητες αυτές είναι οι ακόλουθες:

(I) *Άθροισμα* (sum): Η ιδιότητα αυτή αναφέρεται στην ομαδοποίηση των γραμμών του πίνακα συνεκτικότητας των κορυφών που έχουν το ίδιο πλήθος όρων (δηλαδή, το ίδιο πλήθος μη μηδενικών στοιχείων). Θα πρέπει να σημειωθεί ότι υπολογίζεται το άθροισμα των όρων κάθε γραμμής του πίνακα και στη συνέχεια τα αθροίσματα αυτά ταξινομούνται κατά αύξουσα σειρά, όπως φαίνεται στην τρίτη στήλη του πίνακα 3.3.2-1. Για παράδειγμα, αν και οι γραμμές 1, 3, 10 και 12 έχουν το ίδιο πλήθος όρων, τα αθροίσματα 12, 16, 36 και 40 των όρων αυτών, για κάθε ξεχωριστή γραμμή, ταξινομούνται κατά αύξουσα σειρά.

(II) *Μετρησιμότητα* (ponderation): Η ιδιότητα αυτή αναφέρεται σε ένα διάνυσμα (τέταρτη στήλη του πίνακα 3.3.2-1), το οποίο προκύπτει διαιρώντας κάθε όρο της τρίτης στήλης του πίνακα 3.3.2-1 με το συνολικό πλήθος των όρων της αντίστοιχης γραμμής του πίνακα συνεκτικότητας των κορυφών. Μπορεί κανείς να παρατηρήσει ότι τα επιμέρους στοιχεία του διανύσματος κατατάσσονται κατά αύξουσα σειρά από την πρώτη προς την τελευταία γραμμή.

(III) *Έκταση* (span) : Η τρίτη ιδιότητα αφορά στον υπολογισμό του αθροίσματος του μικρότερου και του μεγαλύτερου στοιχείου κάθε γραμμής. Τα αθροίσματα αυτά αποτελούν τα στοιχεία ενός διανύσματος, τα οποία επίσης κατατάσσονται κατά αύξουσα σειρά (πέμπτη στήλη του πίνακα 3.3.2-1).

Κορυφή (1)	Πίνακας Συννεκτικότητας των Κορυφών (2)	Άθρ/σμα (3)	Μετρ/τα (4)	Έκταση (5)
1	1 2 4 5	12	3	6
2	2 1 3 4 5 6	21	3,5	7
3	3 2 5 6	16	4	8
4	4 1 7 2 5 8	27	4,5	9
5	5 1 2 3 4 6 7 8 9	45	5	10
6	6 2 3 5 8 9	33	5,5	11
7	7 4 5 8 10 11	45	7,5	15
8	8 4 5 6 7 9 10 11 12	72	8	16
9	9 5 6 8 11 12	51	8,5	17
10	10 7 8 11	36	9	18
11	11 7 8 9 10 12	57	9,5	19
12	12 8 9 11	40	10	20

Πίνακας 3.3.2-1: Ιδιότητες Πίνακα Συνεκτικότητας των Κορυφών

Οι ιδιότητες που αναφέρθηκαν, χρησιμοποιούνται σαν κριτήρια από το συγκεκριμένο αλγόριθμο και εξετάζονται σε συνεχή βάση κατά τη διάρκεια των διαφόρων επαναλήψεων που πραγματοποιούνται κατά τη φάση εκτέλεσης. Σε κάθε διαφορετική φάση του αλγόριθμου εξετάζεται ένας αποδοτικός συνδυασμός αυτών των κριτηρίων, έτσι ώστε να ελέγχεται αν ικανοποιούνται κάποιες αναγκαίες συνθήκες. Αναλυτικά, στη διάρκεια των δύο διακεκριμένων φάσεων του αλγόριθμου εκτελούνται οι παρακάτω διαδικασίες :

Φάση 1 : Οι κορυφές επαναδιατάσσονται με βάση τα κριτήρια έκταση και μετρησιμότητα, αντίστοιχα. Η διαδικασία επαναδιάταξης των κορυφών επαναλαμβάνεται, λαμβάνοντας υπόψη αυτά τα δύο κριτήρια, μέχρις ότου καταστεί ανέφικτη οποιαδήποτε περαιτέρω μείωση του εύρους ζώνης.

Φάση 2 : Μετά από την ολοκλήρωση της πρώτης φάσης, οι κορυφές επαναδιατάσσονται ξανά έτσι, ώστε να ικανοποιούνται τα κριτήρια μετρησιμότητας και αθροίσματος, ταυτόχρονα. Η διαδικασία αυτή επίσης

επαναλαμβάνεται μέχρις ότου να μην είναι δυνατή κάποια περαιτέρω μείωση του εύρους ζώνης. Συνήθως, στο σημείο αυτό επανεκτελείται η πρώτη φάση του αλγόριθμου με σκοπό την πιθανή επίτευξη κάποιας περαιτέρω μείωσης του εύρους ζώνης.

Το βασικό πλεονέκτημα αυτού του αλγόριθμου, σε σχέση με τον προηγούμενο αλγόριθμο, έγκειται στο ότι απαιτείται μόνον η δημιουργία του πίνακα συνεκτικότητας των κορυφών και όχι ο ακριβής προσδιορισμός του πλήθους των μη μηδενικών στοιχείων του αραιού πίνακα. Επιπλέον, η όλη διαδικασία ολοκληρώνεται με τις ελάχιστες δυνατές απαιτήσεις σε ότι αφορά σε αποθηκευτικό χώρο, αλλά και σε υπολογιστικό χρόνο, γεγονός που οφείλεται στη χρήση απλών μαθηματικών και λογικών εντολών μόνο.

3.3.3 Χωρο-αποδοτικές (Area Efficient) Συστολικές Σχεδιάσεις

Οι αλγόριθμοι που έχουν συζητηθεί έως τώρα, επιδιώκουν τη διαμόρφωση του πλέον κατάλληλου πίνακα μετάθεσης P, με βάση την αρχική μορφή του συντελεστή πίνακα A. Όπως ήδη αναφέρθηκε, η επιλογή του πίνακα P είναι δυνατό να επηρεάσει αποφασιστικά το ποσοστό των μη μηδενικών στοιχείων που μπορεί να προκύψουν κατά τη διαδικασία παραγοντοποίησης του A. Επιπλέον, οι πίνακες των συστημάτων στα οποία αναφερθήκαμε, θεωρήσαμε ότι είναι αραιοί και πολύ μεγάλου μεγέθους, αλλά όχι συγκεκριμένου τύπου.

Μία εντελώς διαφορετική προσέγγιση αφορά σε συντελεστές πίνακες συγκεκριμένου τύπου οι οποίοι προκύπτουν από ελλειπτικές και παραβολικές εξισώσεις δευτέρου και τρίτου βαθμού. Πιο συγκεκριμένα, όταν το n είναι μεγάλο, οι πίνακες αυτοί εμφανίζονται ως αραιοί ταινιωτοί (banded) πίνακες στους οποίους, ακόμη και κατά μήκος της ταινίας, η διάταξη των στοιχείων παρουσιάζει, επίσης, αραιή δομή. Στις επόμενες Παραγράφους, χωρίς μείωση της γενικότητας, και για την παρουσίαση της συγκεκριμένης προσέγγισης, θα χρησιμοποιηθούν ως παράδειγμα συντελεστές πίνακες που προκύπτουν από δευτεροβάθμιες, μόνο, παραβολικές και ελλειπτικές εξισώσεις.

Η παραβολική εξίσωση,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad (x,y) \in R \equiv (0,1)\,x(0,1) \tag{3.3.3:1}$$

με οριακές συνθήκες u(x,y)=0, παράγει ένα σύστημα το οποίο βασίζεται στον τύπο των πεπερασμένων διαφορών των πέντε σημείων,

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0, \qquad (3.3.3:2)$$

με μεσοδιαστήματα πλέγματος $\Delta x/\Delta y$ στις κατευθύνσεις x, y, αντίστοιχα, και σημεία πλέγματος $u(i\Delta x, j\Delta y) = u_{i,j}$. Αν υπάρχουν n^2 εσωτερικά σημεία, τότε μία αρίθμηση των σημείων αυτών κατά στήλη μας δίνει έναν πραγματικό πενταδιαγώνιο πίνακα μεγέθους n^2, ο οποίος έχει την παρακάτω μορφή:

$$A = \begin{bmatrix} A_1 & -I & & & \\ -I & \ddots & \ddots & & O \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ O & & & -I & A_n \end{bmatrix} \quad \text{και} \quad A_i = \begin{bmatrix} 4 & 1 & & & \\ 1 & \ddots & \ddots & & O \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ O & & & 1 & 4 \end{bmatrix} \qquad (3.3.3:3)$$

όπου τα διανύσματα x και y έχουν μέγεθος $(n^2 x 1)$ και αντιστοιχούν στα διανύσματα των αγνώστων και των σταθερών όρων, αντίστοιχα, του συστήματος (3.1:1). Το διάνυσμα y δομείται με βάση τις οριακές συνθήκες του προβλήματος.

Κατά την LU-παραγοντοποίηση τέτοιου είδους πινάκων μη μηδενικά στοιχεία δημιουργούνται και τοποθετούνται σε θέσεις με μηδενικά στοιχεία του κάτω τριγωνικού πίνακα L. Το γεγονός αυτό συνεπάγεται τη χρήση συστολικών αρχιτεκτονικών με ένα αρκετά μεγάλο πλήθος κελιών. Ο βασικός σκοπός της συγκεκριμένης προσέγγισης είναι να περιορίσει το πλήθος των μη μηδενικών στοιχείων τα οποία δημιουργούνται κατά τη διαδικασία της παραγοντοποίησης σε συγκεκριμένες διαγωνίους του πίνακα. Κάτι τέτοιο συνεπάγεται μία *προσεγγιστική* (approximate) διαδικασία παραγοντοποίησης κατά τη διάρκεια της οποίας προσδιορίζεται μία ακολουθία τιμών r_i , i=1,...,k, η οποία υποδηλώνει το πλήθος των μη μηδενικών διαγωνίων που πρόκειται να διατηρηθούν για τον περιορισμό των δημιουργούμενων μη μηδενικών στοιχείων. Στην περίπτωση αυτή, ελέγχοντας τις παραμέτρους r_i, η ακρίβεια της παραγοντοποίησης κλιμακώνεται σε σχέση με το πλήθος των απαιτούμενων συστολικών κελιών και κατ' επέκταση σε σχέση με τον απαιτούμενο φυσικό χώρο που καταλαμβάνουν αυτά τα κελιά.

3.3.3.1 Πρώτο Επιπέδο Συμπύκνωσης

Έστω ότι χρησιμοποιούμε μία εξαγωνικο-συνδεδεμένη συστολική διάταξη για την παραγοντοποίηση του αραιού ταινιωτού πίνακα A που προκύπτει από την εξίσωση 3.3.3:1. Με W_s και W_i θα συμβολίζεται, αντίστοιχα, το ημι-εύρος ζώνης (semi-bandwidth) και το εσωτερικό εύρος ζώνης του πίνακα A. Παρατηρώντας προσεκτικά τα υπολογιστικά βήματα της διαδικασίας της LU-παραγοντοποίησης σε μία εξαγωνικό-συνδεδεμένη συστολική διάταξη (Bekakos, et al, [14]) προκύπτουν οι παρακάτω πολύ σημαντικές παρατηρήσεις:

(I) Όλα τα στοιχεία του πίνακα A, που ανήκουν σε μία συγκεκριμένη διαγώνιό του, εισέρχονται στο ίδιο κελί της συστολικής διάταξης.

(II) Ο υπολογισμός των στοιχείων μιας διαγωνίου των πινάκων L ή U, πραγματοποιείται μόνο από τα κελιά που βρίσκονται κατα μήκος μιας συγκεκριμένης στήλης της συστολικής διάταξης.

(III) Ένα μη μηδενικό στοιχείο μπορεί να παραχθεί και να τοποθετηθεί σε μία αρχικά μηδενική διαγώνιο, μόνο από εκείνα τα κελιά τα οποία ανήκουν στη στήλη της συστολικής διάταξης στην οποία εισέρχονται τα στοιχεία της διαγωνίου αυτής.

Αν, λοιπόν, απαγορευθεί η τοποθέτηση μη μηδενικών στοιχείων σε μία αρχικά μηδενική διαγώνιο, τότε κανένα από τα κελιά της αντίστοιχης στήλης της συστολικής διάταξης δε θα παράγει κάποιο μη μηδενικό εσωτερικό γινόμενο. Συνεπώς, τα κελιά της συγκεκριμένης στήλης της συστολικής διάταξης μπορούν να αντικατασταθούν από ένα σύνολο *καταχωρητών καθυστέρησης* (delay registers). Κάτι τέτοιο φαίνεται στο σχήμα 3.3.3.1-1. Κατ' επέκταση, η είσοδος και η έξοδος μιας στήλης της συστολικής διάταξης που περιέχει καταχωρητές καθυστέρησης είναι πάντοτε μηδενική και για το λόγο αυτό δεν απαιτούνται συνδέσεις επικοινωνίας των στηλών αυτών με τον κεντρικό υπολογιστή (host machine). Έτσι, μειώνεται το συνολικό πλήθος των συνδέσεων εισόδου/εξόδου, το οποίο ισούται με το πλήθος των μη μηδενικών διαγωνίων του πίνακα A, συν το πλήθος εκείνων των αρχικά μηδενικών διαγωνίων που επιλέγονται για τον περιορισμό των δημιουργούμενων μη μηδενικών στοιχείων.

Το πλήθος των συστολικών κελιών τα οποία εξοικονομούνται κάνοντας χρήση της προσεγγιστικής αυτής διαδικασίας παραγοντοποίησης, προκύπτει εύκολα με τον ακόλουθο τρόπο. Το πλήθος των *πραγματικών* κελιών, C, δίνεται από τη σχέση,

$$C = C_1 + 2 * C_2 \qquad (3.3.3.1:1)$$

με

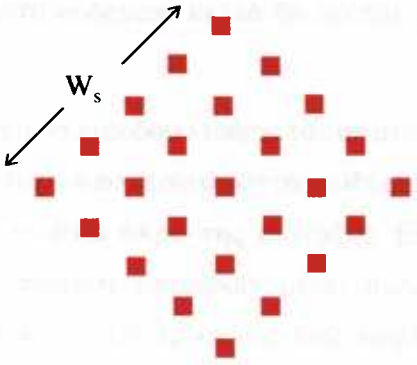
$$C_1 = W_s + 2 * \sum_{i=1}^{\lfloor \frac{W_i - 1}{2} \rfloor} (W_s - i) = W_s + 2W_s * \left\lfloor \frac{W_i - 1}{2} \right\rfloor - \left\lfloor \frac{W_i - 1}{2} \right\rfloor * \left(\left\lfloor \frac{W_i - 1}{2} \right\rfloor + 1 \right) \qquad (3.3.3.1:2)$$

και

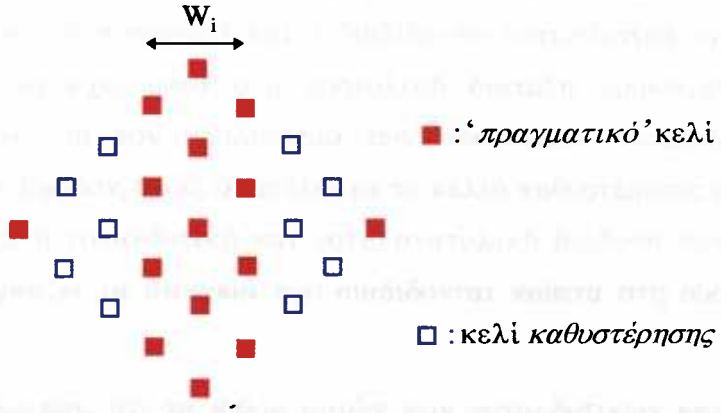
$$C_2 = \sum_{i=1}^{r} \{ (W_s - t + 1) - i + 1 \} = (W_s - t + 2)r - \frac{r}{2}(r+1), \qquad (3.3.3.1:3)$$

όπου r είναι το πλήθος των ταινιών από στοιχεία του πίνακα οι οποίες διατηρούνται και t είναι η θέση της πρώτης τέτοιας ταινίας. Η εξοικονόμηση των συστολικών κελιών, η οποία είναι ανάλογη του πλήθους των καταχωρητών καθυστέρησης, δίνεται από τη σχέση,

$$S = W_s^2 - C. \qquad (3.3.3.1:4)$$



α) Αρχική Εξαγωνικο-συνδεδεμένη Συστολική Διάταξη



β) Συμπυκνωμένη Εξαγωνικο-συνδεδεμένη Συστολική Διάταξη (m=5, r=1)

Σχήμα 3.3.3.1-1 : Αντικατάσταση Συστολικών Κελιών και Συμπύκνωση της Συστολικής Διάταξης

3.3.3.2 Δεύτερο Επίπεδο Συμπύκνωσης

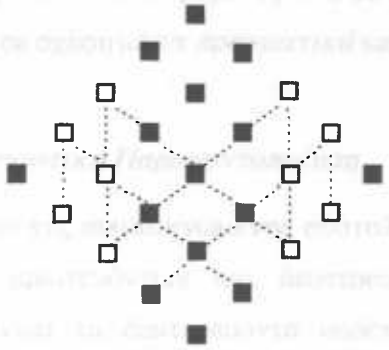
Είναι προφανές ότι η εφαρμογή της συγκεκριμένης διαδικασίας παραγοντοποίησης, κατά την οποία απαλείφονται μερικές διαγώνιοι, επιτρέπει σε εκείνες τις στήλες των συστολικών κελιών, στις οποίες εισέρχονται αυτές οι διαγώνιοι, να αντικατασταθούν με *κελιά καθυστέρησης* (delay cells). Τα κελιά αυτά ονομάζονται *πρωτεύοντα ουδέτερα κελιά* (primary neutral cells). Από τη ροή των δεδομένων, μετά από την εφαρμογή της συμπύκνωσης πρώτου επιπέδου, διαπιστώνεται ότι υπάρχουν επιπλέον κελιά στη συστολική διάταξη, εκτός από αυτά που έχουν ήδη προσδιοριστεί, τα οποία παράγουν, επίσης, μηδενικά εσωτερικά γινόμενα. Ένα δεύτερο επίπεδο συμπύκνωσης αντικαθιστά αυτά τα κελιά, τα οποία ονομάζονται *δευτερεύοντα ουδέτερα κελιά* (secondary neutral cells), με επιπλέον καταχωρητές καθυστέρησης. Προκειμένου, λοιπόν, να προσδιοριστούν αυτά τα δευτερεύοντα ουδέτερα κελιά θα πρέπει να ληφθούν υπόψη τα παρακάτω χαρακτηριστικά:

(I) Όταν ένα δεδομένο εισόδου εισέρχεται σε μία στήλη της συστολικής διάταξης που αποτελείται από πρωτεύοντα ουδέτερα κελιά, τότε αυτό διαδίδεται κάθετα προς το άνω άκρο της διάταξης. Είναι προφανές, ότι μετά από τη συμπύκνωση πρώτου επιπέδου, μόνο μηδενικά στοιχεία εισέρχονται σε στήλες με τέτοιου είδους κελιά, ενώ καμία τροποποίηση των μηδενικών αυτών τιμών δεν υφίσταται κατά τη διάρκεια της κάθετης διάδοσής τους.

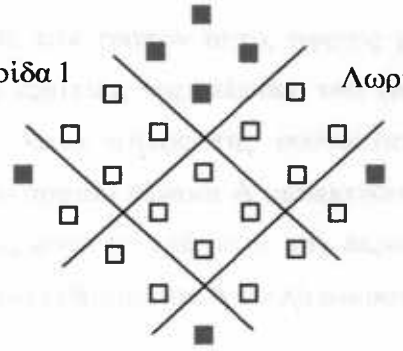
(II) Κατά τη διάρκεια της διαδικασίας παραγοντοποίησης τα δημιουργούμενα στοιχεία l_{ij} και u_{ij} των πινάκων L και U, διαδίδονται νοτιο-δυτικά και νοτιο-ανατολικά, αντίστοιχα, μέσα στη συστολική διάταξη προκειμένου να χρησιμοποιηθούν για τον υπολογισμό των υπολοίπων στοιχείων των πινάκων αυτών. Εφόσον, όμως, οι στήλες με τα κελιά καθυστέρησης εξάγουν μηδενικές τιμές, η νοτιο-δυτική και νοτιο-ανατολική διάδοση των τιμών αυτών δεν επηρεάζει τα στοιχεία που διαδίδονται κάθετα στη συστολική διάταξη.

Είναι προφανές, λοιπόν, ότι τα κελιά αυτών των νοτιο-δυτικών και νοτιο-ανατολικών μονοπατιών, από τα οποία διέρχονται μηδενικές τιμές, πραγματοποιούν μη χρήσιμους υπολογισμούς. Επομένως, τα κελιά αυτά μπορούν να θεωρηθούν σαν δευτερεύοντα ουδέτερα κελιά, τα οποία επίσης αντικαθίστανται από καταχωρητές καθυστέρησης. Η συμπύκνωση δευτέρου επιπέδου της συστολικής

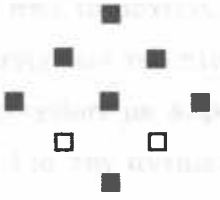
διάταξης περιγράφεται στο σχήμα 3.3.3.2-1. Μετά από την εφαρμογή αυτής της διαδικασίας προκύπτει ότι το πλήθος των συστολικών κελιών είναι πλέον ανάλογο του συνόλου των μη μηδενικών διαγωνίων του αραιού πίνακα Α, συν το πλήθος των διαγωνίων που επιλέγονται να χρησιμοποιηθούν για τον περιορισμό των μη μηδενικών στοιχείων (fill-ins).



α) Προσδιορισμός *Δευτερευόντων Ουδέτερων Κελιών*



β) Λωρίδες (slices) Συγχρονισμού και Απαλοιφή Καθυστερήσεων



γ) Συμπυκνωμένη Εξαγωνικο-συνδεδεμένη Συστολική Διάταξη

Σχήμα 3.3.3.2-1: Κατασκευή της Συστολικής Διάταξης για την Υλοποίηση της Προσεγγιστικής Παραγοντοποίησης

Παρόλο που μετά από την εφαρμογή και του δεύτερου επιπέδου συμπύκνωσης τα *πραγματικά* κελιά της χρησιμοποιούμενης συστολικής διάταξης έχουν μειωθεί

σημαντικά, η συνολική χρονική πολυπλοκότητα που απαιτείται για την υλοποίηση της προσεγγιστικής διαδικασίας παραγοντοποίησης, παραμένει αμετάβλητη. Αυτό συμβαίνει γιατί αν και οι καταχωρητές καθυστέρησης ουσιαστικά δεν πραγματοποιούν χρήσιμους υπολογισμούς, παραμένει υπαρκτός ο χρόνος υλοποίησης αυτών των υπολογισμών. Κατά συνέπεια, η εξοικονόμηση του φυσικού χώρου που προκύπτει οφείλεται στο γεγονός ότι οι καταχωρητές καθυστέρησης απαιτούν λιγότερο χώρο σε σχέση με τα *πραγματικά* κελιά.

3.3.3.3 Συστολική Προσεγγιστική Παραγοντοποίηση

Η απαλοιφή των λωρίδων της συμπυκνωμένης συστολικής διάταξης συνεπάγεται την απομάκρυνση των πρωτευόντων και δευτερευόντων ουδέτερων κελιών. Ωστόσο, όταν απαλείφονται τα δευτερεύοντα ουδέτερα κελιά δημιουργούνται ακμές, σε μία μικρότερη εξαγωνικο-συνδεδεμένη συστολική αρχιτεκτονική, οι οποίες επιτρέπουν την άμεση επικοινωνία σε διαφορετικά τμήματα της αρχικής συστολικής διάταξης. Με τον τρόπον αυτό, αφενός μεν μειώνεται η συνολική χρονική πολυπλοκότητα εξαιτίας της μείωσης του χρόνου που απαιτεί πλέον η διάδοση ενός στοιχείου κατά μήκος της συστολικής διάταξης, αφετέρου οι εξωτερικές διαγώνιοι του αραιού πίνακα A μετακινούνται πιο κοντά στην κύρια διαγώνιό του. Το γεγονός αυτό συνεπάγεται την παραγοντοποίηση, τελικά, ενός πίνακα ο οποίος έχει ίδιο μεγέθος με τον A, αλλά μικρότερο εύρος ζώνης.

Στην περίπτωση αυτή προκύπτουν δύο σημαντικά προβλήματα. Το πρώτο αφορά στο ότι πλέον παραγοντοποιείται ένας πίνακας, ο οποίος αναφέρεται σε ένα τελείως διαφορετικό πρόβλημα από το αρχικό, ενώ το δεύτερο αφορά στο ότι η μετακίνηση των εξωτερικών διαγωνίων του πίνακα A, προς την κύρια διαγώνιό του, προκαλεί την εμφάνιση στοιχείων με απροσδιόριστες τιμές στα άκρα των εξωτερικών διαγωνίων του A. Για την αντιμετώπιση αυτών των προβλημάτων πραγματοποιείται, από την αρχή, ένας μετασχηματισμός του αρχικού πίνακα A, ο οποίος εφαρμόζεται επίσης και στους πίνακες L και U που προκύπτουν από την παραγοντοποίησή του. Η διαδικασία μετασχηματισμού περιγράφεται συνοπτικά με τα παρακάτω βήματα:

(I) Διαμόρφωσε έναν νέο πίνακα A′, απαλείφοντας τις διαγωνίους του A που δεν πρόκειται να χρησιμοποιηθούν για την τοποθέτηση μη μηδενικών στοιχείων, που πιθανώς να προκύψουν κατά τη διαδικασία

παραγοντοποίησής του. Κατόπιν μετακίνησε τις εξωτερικές διαγωνίους, που θα χρησιμοποιηθούν για το σκοπό αυτό, προς την κύρια διαγώνιο του A. Υλοποίησε την παραγοντοποίηση του A′ για να ληφθούν οι πίνακες L′ και U′.

(II) Τροποποίησε τους πίνακες L′ και U′, ώστε να προκύψουν οι αντίστοιχοι αραιοί πίνακες L″ και U″, μέσω της μετατόπισης, στις αρχικές τους θέσεις, των εξωτερικών διαγωνίων που χρησιμοποιήθηκαν, αγνοώντας όλους τους όρους οι οποίοι καταλήγουν εκτός των διαστάσεων των πινάκων μετά από τη μετατόπιση.

Στα στοιχεία με απροσδιόριστες τιμές, τα οποία προκύπτουν στα άκρα των εξωτερικών διαγωνίων που μετατοπίζονται, εκχωρούνται οι τιμές 0 και -1, αν και θα πρέπει να σημειωθεί ότι η επιλογή αυτών των τιμών είναι αποφασιστικής σημασίας για την ποιότητα των αποτελεσμάτων της προσεγγιστικής παραγοντοποίησης.

Η διαδικασία που περιγράφηκε παρουσιάζει δύο βασικά πλεονεκτήματα. Πρώτον, το σύστημα που προκύπτει έχει παρόμοια δομή με αυτή του αρχικού συστήματος και, δεύτερον, τα μη μηδενικά στοιχεία που εμφανίζονται κατά τη διάρκεια της παραγοντοποίησης, και αφορούν στα στοιχεία με απροσδιόριστες τιμές των μετατοπιζόμενων διαγωνίων, μετατίθενται εκτός πίνακα εξαλείφοντας έτσι την επίδραση που ασκούν στην τελική λύση.

3.4 Συμπεράσματα

Οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται σήμερα για την ελαχιστοποίηση του εύρους ζώνης και της κατατομής ενός αραιού πίνακα είναι ευρετικοί (heuristic) και δεν εγγυώνται τη βέλτιστη δυνατή ελαχιστοποίηση. Όπως είδαμε, διάφορες εναλλακτικές προσεγγίσεις έχουν προταθεί με κάθεμία από αυτές να βελτιώνει κάποια παράμετρο σε βάρος, πιθανώς, κάποιας άλλης.

Το κοινό σημείο αναφοράς όλων των αλγορίθμων είναι η προσπάθεια μείωσης, τόσο του χώρου αποθήκευσης που θα καταλαμβάνει, τελικά, ο χρησιμοποιούμενος συντελεστής πίνακας, όσο και του πλήθους των αλγεβρικών πράξεων που απαιτούνται για την επίλυση του εκάστοτε συγκεκριμένου συστήματος. Όσον αφορά στην περίπτωση του φασματικού αλγόριθμου, παρατηρείται ότι ενώ απαιτεί

σημαντικά μεγαλύτερη συνολική χρονική πολυπλοκότητα για τη μείωση του φακέλλου, επιτυγχάνει μία σημαντική μείωση του αντίστοιχου χρόνου, στο βαθμό που υλοποιείται η παραγοντοποίηση των συμπυκνωμένων αραιών πινάκων που προκύπτουν.

Παρόμοια αποτελέσματα, όσον αφορά στο χρόνο παραγοντοποίησης, επιτυγχάνονται και μέσω της διαδικασίας προσεγγιστικής παραγοντοποίησης στην οποία ο συντελεστής πίνακας που χρησιμοποιείται έχει συγκεκριμένη μορφή. Η προσέγγιση αυτή περιορίζει, ουσιαστικά, το πλήθος των μη μηδενικών στοιχείων που δημιουργούνται κατά τη διαδικασία της παραγοντοποίησης, ενώ ταυτόχρονα οδηγεί στη χρήση χωρο-αποδοτικών εξαγωνικο-συνδεδεμένων συστολικών διατάξεων που υλοποιούν την παραγοντοποίηση αυτή. Ωστόσο, η κρισιμότητα της μεθόδου έγκειται στην κατάλληλη επιλογή του πλήθους των διαγωνίων που πρόκειται να διατηρηθούν κατά την παραγοντοποίηση, έτσι ώστε να είναι αποδεκτό το προσεγγιστικό λάθος (approximation error) που προκύπτει. Διάφορα πειραματικά αποτελέσματα έχουν δείξει ότι η μέθοδος επιδεικνύει ραγδαία ελαχιστοποίηση του προσεγγιστικού λάθους για μικρό πλήθος διαγωνίων.

Τέλος, οι σειριακοί αλγόριθμοι συμπύκνωσης των Rosen, [63] και Akhras, et al, [1] παρουσιάζουν σχετικά ικανοποιητικά αποτελέσματα, όσον αφορά στην ελαχιστοποίηση του εύρους ζώνης ενός αραιού συμμετρικού πίνακα, αν και αυτά εξαρτώνται πάντοτε από την αρχική διάταξη των μη μηδενικών στοιχείων του. Το πλεονέκτημά τους αφορά κυρίως στο ότι εμπεριέχουν απλές λειτουργίες σύγκρισης και αντιμετάθεσης, οι οποίες είναι σαφώς λιγότερο πολύπλοκες από αυτές που απαιτεί η ελαχιστοποίηση του εύρους ζώνης κάνοντας χρήση της θεωρίας των γράφων που θα περιγραφεί στο επόμενο Κεφάλαιο.

Κεφάλαιο 4

Ελαχιστοποίηση του Εύρους Ζώνης Αραιών Πινάκων σε Συστολικές Αρχιτεκτονικές

4.1 Εισαγωγή

Ο χώρος της θεωρίας των γράφων έχει αποδειχθεί ότι προσφέρει ένα από τα πλέον αξιόλογα εργαλεία για την περιγραφή και το χειρισμό πολλών και σημαντικών θεμάτων τα οποία σχετίζονται με το χώρο των υπολογιστών. Η συμβολή του ήταν, και εξακολουθεί να είναι, πολύ σημαντική εξαιτίας του ότι επιτρέπει τη δυναμική προσέγγιση του προβλήματος με την απεικόνισή του με γράφημα το οποίο αναπαριστά την αλγοριθμική ροή της πληροφορίας μέσω της δομής των διεργασιών που μετασχηματίζουν την πληροφορία εισόδου σε αποτελέσματα εξόδου και της τοπολογίας των φυσικών επεξεργαστών οι οποίοι εκτελούν ουσιαστικά αυτές τις διεργασίες. Στο βαθμό που μία αρχιτεκτονική αναπαρίσταται ως ένα δίκτυο από επεξεργαστές, το μείζον θέμα είναι η βέλτιστη δυνατή αντιστοίχιση των γραφημάτων που περιγράφουν τη ροή των δεδομένων και την τοπολογία των επεξεργαστών. Για την αντιμετώπιση αυτού του δύσκολου προβλήματος καταφεύγουμε κυρίως σε ευρετικές (heuristic) τεχνικές, ακόμη και σήμερα, οι οποίες, όμως, όχι μόνο δεν εγγυώνται βέλτιστες λύσεις, αλλά συχνά παρουσιάζουν πλήρη αδυναμία στο να παρέχουν αποδεκτές, τουλάχιστον, λύσεις.

Από την άλλη μεριά, ένα ικανό πλήθος πολύπλοκων και δυναμικά εξελισσόμενων προβλημάτων προσεγγίζεται ικανοποιητικά μέσω της θεωρίας των γράφων. Η πολυμορφία των ιδιοτήτων που διαθέτουν τα γραφήματα, αφενός μεν παρέχει τη δυνατότητα της μελέτης πολλών διαφορετικών παραγόντων, οι οποίοι μπορεί να οδηγήσουν σε κάποια λύση, αφετέρου τα καθιστά ιδιαίτερα δύσχρηστα καθώς, συχνά, αυξάνεται σημαντικά η πολυπλοκότητα στο χειρισμό του ίδιου του προβλήματος. Ένα από τα προβλήματα εκείνα τα οποία έχουν προσεγγιστεί ικανοποιητικά κάνοντας χρήση της θεωρίας των γράφων είναι και το πρόβλημα της ελαχιστοποίησης του εύρους ζώνης αραιών πινάκων μεγάλου μεγέθους.

Όσον αφορά, λοιπόν, στη θεωρία των γράφων, θα πρέπει να σημειωθεί ότι οι μεταθέσεις των γραμμών ενός πίνακα, αντιστοιχούν στην επαναρίθμηση των κορυφών ενός γραφήματος G. Αντιπροσωπευτικοί αλγόριθμοι αυτής της κατηγορίας είναι αυτοί που προτάθηκαν από τους (Cuthill-Mckee, [31] και Gibbs, et al, [39]) και οι οποίοι χρησιμοποιούν έννοιες της θεωρίας των γράφων για την επαναρίθμηση των κορυφών του γραφήματος, το οποίο αντιστοιχεί στον πίνακα γειτνίασης ο οποίος σχετίζεται με τον υπό εξέταση συντελεστή πίνακα. Η έννοια της *δομής επιπέδων* (level structure), η οποία αφορά στη διάταξη (ordering) των κορυφών ενός γραφήματος σε επίπεδα, είναι άμεσα συνυφασμένη με την τεχνική που χρησιμοποιούν οι συγκεκριμένοι αλγόριθμοι.

Στη συνέχεια θα παρουσιαστεί μία συγκριτική μελέτη των αλγορίθμων που αναφέρθηκαν σε σχέση με την προσέγγιση των (Bekakos et al., [11,12]), η οποία προτείνει τη χρήση μιας *δυναμικά αναδιαρθρούμενης* (reconfigurable) συστολικής αρχιτεκτονικής για την εύρεση όλων των δυνατών δομών επιπέδων, με σκοπό την επιλογή εκείνης που μετά βεβαιότητος θα επιτυγχάνει την καλύτερη δυνατή μείωση του εύρους ζώνης και της κατατομής του αρχικού αραιού πίνακα. Η προσέγγιση αυτή, στις περισσότερες περιπτώσεις, παρέχει καλύτερα αποτελέσματα όσον αφορά στη μείωση του εύρους ζώνης και της κατατομής, ενώ τις περισσότερες φορές η απαιτούμενη συνολική χρονική πολυπλοκότητα πειραματικά αποδείχθηκε της ίδιας τάξης με αυτήν που απαιτείται από τον αλγόριθμο των Cuthill-Mckee.

4.2 Θεωρία Γράφων: Βασικές Έννοιες και Ορισμοί

Ένα γράφημα G(V,E) είναι ένα *διατεταγμένο ζεύγος*, όπου το V είναι ένα πεπερασμένο σύνολο τα στοιχεία του οποίου καλούνται *κορυφές*, και το E είναι ένα σύνολο από μη διατεταγμένα ζεύγη διακεκριμένων κορυφών του V. Κάθε στοιχείο $\{v,u\} \in E$, (με $v,u \in E$) καλείται *ακμή*, η οποία συνδέει τις κορυφές v και u. Για μία ακμή $e=\{v,u\} \in E$, οι κορυφές v και u χαρακτηρίζονται ως *προσκείμενες* (incident) στην e και *γειτονικές* (adjacent) μεταξύ τους. Ο *βαθμός* (degree) μιας κορυφής v ορίζεται ως το πλήθος των ακμών που πρόσκεινται στην κορυφή v και συμβολίζεται με $d_v(G)$.

Ένας *περίπατος* (walk) μήκους k σε ένα γράφημα G είναι μία διαδοχή, της μορφής uv, vw, wx, ..., yz, από k ακμές του G. Αν όλες οι ακμές ενός περίπατου, αλλά όχι απαραίτητα όλες οι κορυφές, είναι διαφορετικές, τότε ο περίπατος καλείται *ίχνος* (trail). Επιπλέον, αν και όλες οι κορυφές είναι διαφορετικές τότε το ίχνος καλείται *μονοπάτι* (path). Ένα γράφημα G καλείται *συνεκτικό* (connected) αν μεταξύ οποιουδήποτε ζεύγους κορυφών του υπάρχει ένα μονοπάτι που να τις συνδέει, διαφορετικά το γράφημα καλείται *μη συνεκτικό* (disconnected). Κάθε μη συνεκτικό γράφημα αποτελείται από ένα σύνολο συνεκτικών υπογραφημάτων (subgraphs) τα οποία καλούνται *συνιστώσες* (components) του γραφήματος. Η *απόσταση* μεταξύ των κορυφών u και v ενός συνεκτικού γραφήματος ισούται με το μήκος του συντομότερου μονοπατιού που συνδέει τις κορυφές αυτές. Ως *διάμετρος* (diameter) του γραφήματος G ορίζεται το συντομότερο μονοπάτι που συνδέει τις δύο πλέον απομακρυσμένες κορυφές του, ενώ ο ίδιος όρος χρησιμοποιείται και για την έκφραση του μήκους ενός τέτοιου μονοπατιού.

Μία πολύ σημαντική έννοια, η οποία χρησιμοποιείται σε πολλούς αλγόριθμους Ελαχιστοποίησης του Εύρους Ζώνης (EEZ_α) αραιών πινάκων είναι αυτή της *δομής επιπέδων*. Μία δομή επιπέδων LS(G), ενός γραφήματος G, είναι μία διαμέριση (partition) του συνόλου των κορυφών του σε επίπεδα $L_1, L_2, ..., L_k$, έτσι ώστε,

(I) όλες οι κορυφές που είναι γειτονικές με τις κορυφές του επιπέδου L_1 να βρίσκονται είτε στο επίπεδο L_1, ή στο επίπεδο L_2,

(II) όλες οι κορυφές που είναι γειτονικές με τις κορυφές του επιπέδου L_k να βρίσκονται είτε στο επίπεδο L_{k-1}, ή στο επίπεδο L_k,

(III) όλες οι κορυφές που είναι γειτονικές με τις κορυφές του επιπέδου L_i, για $1< i < k$, να βρίσκονται είτε στα επίπεδα L_{i-1}, L_i, ή στο επίπεδο L_{i+1}.

Σε κάθε κορυφή $v \in V$ αντιστοιχεί μία συγκεκριμένη δομή επιπέδων $LS_v(G)$, η οποία καλείται δομή επιπέδων με ρίζα την κορυφή v. Τα επίπεδα μιας τέτοιας δομής ορίζονται ως εξής:

(I) $L_1 = \{ v \}$, και

(II) για $i > 1$, L_i είναι το σύνολο όλων εκείνων των κορυφών οι οποίες είναι γειτονικές με τις κορυφές του επιπέδου L_{i-1} και δεν έχουν ακόμη εκχωρηθεί σε κάποιο επίπεδο.

Σε οποιαδήποτε δομή επιπέδων LS(G), με ή χωρίς ρίζα, το *εύρος* (width) του επιπέδου i ορίζεται ως $w_i(L)=| L_i |$ (δηλαδή, ο πληθάριθμος του συνόλου L_i), ενώ το εύρος της δομής ορίζεται ως $w(L)=\max\{ w_i \}$. Επίσης για οποιαδήποτε δομή επιπέδων, ισχύει ότι μία αρίθμηση f_i του γραφήματος G, εκχωρεί διαδοχικούς θετικούς ακέραιους στις κορυφές της δομής, από επίπεδο σε επίπεδο, ξεκινώντας από το πρώτο επίπεδο και καταλήγοντας στο τελευταίο.

Ένας (nxn) αραιός πίνακας, $A=(\alpha_{i,j})$, μπορεί να θεωρηθεί σαν ένα μη κατευθυνόμενο γράφημα $G=(V,E)$, στο οποίο το σύνολο V έχει *n* κορυφές, $\{ v_1, v_2, \ldots, v_n \}$, ενώ ισχύει ότι $\{v_i, v_j\} \in E$, αν $\alpha_{i,j} \neq 0$ και $i \neq j$. Κατά συνέπεια, το πρόβλημα της ελαχιστοποίησης του εύρους ζώνης και της κατατομής μπορεί να επαναπροσδιοριστεί ως εξής: Επαναρίθμησε τις κορυφές του γραφήματος G, χρησιμοποιώντας μία ένα-προς-ένα απεικόνιση, g, από το σύνολο V στο σύνολο $\{1,2,\ldots,n\}$ έτσι ώστε να ελαχιστοποιηθεί η τιμή της παραμέτρου

$$\beta = \max\{ |g(i)-g(j)| : \alpha_{i,j} \neq 0, i \neq j, i,j=1,\ldots,n\} \qquad (4.2:1)$$

η οποία, ουσιαστικά εκφράζει το εύρος ζώνης του πίνακα A.

4.3 Σειριακοί Αλγόριθμοι Ελαχιστοποίησης του Εύρους Ζώνης και της Κατατομής ενός Αραιού Πίνακα

Ο πλέον διαδεδομένος αλγόριθμος ελαχιστοποίησης του εύρους ζώνης και της κατατομής ενός αραιού πίνακα είναι ο ανάστροφος αλγόριθμος των Cuthill-Mckee

(Reverse Cuthill-Mckee algorithm - RCM). Ο αλγόριθμος αυτός διακρίνεται σε τέσσερις φάσεις, οι οποίες συνοπτικά μπορούν να περιγραφούν ως εξής:

Φάση 1: Δημιούργησε τις δομές επιπέδων με ρίζα κάθε κορυφή που έχει βαθμό μικρότερο ή ίσο από το LD (Low Degree), όπου

$$LD \leq \max\{\min\{(d_{max}+d_{min})/2, d_{median}-1\}, d_{min}\} \qquad (4.3:1)$$

Φάση 2: Σε κάθε δομή επιπέδων που προέκυψε εκχώρησε διαδοχικούς θετικούς ακέραιους, από επίπεδο σε επίπεδο, δηλαδή κάνε *επαναρίθμηση* (re-numbering) των κορυφών του γραφήματος.

Φάση 3: Για κάθε αρίθμηση f, υπολόγισε το αντίστοιχο εύρος ζώνης και κατόπιν επέλεξε την αρίθμηση που παράγει το μικρότερο εύρος ζώνης.

Φάση 4: Αντίστρεψε την αρίθμηση θέτοντας $i = n-i+1$, για $i=1,...,n$.

Η τέταρτη φάση του αλγόριθμου προτάθηκε έπειτα από την παρατήρηση ότι η κατατομή, συνήθως, μειώνεται περαιτέρω αν οι κορυφές αριθμηθούν κατά φθίνουσα σειρά, δηλαδή από το n προς το 1, αντί κατά αύξουσα, δηλαδή από το 1 προς το n. Επιπλέον, αποδείχτηκε ότι η τροποποίηση αυτή, τελικά, δεν είναι δυνατό σε καμία περίπτωση να αυξήσει την τιμή της κατατομής του αραιού πίνακα, ενώ δεν έχει καμία επίπτωση στην τιμή του εύρους ζώνης αυτού.

Μία αξιόλογη προσπάθεια βελτίωσης αυτού του αλγόριθμου έγινε από τους Gibbs, Poole & Stockmayer (Gibbs, et al, [39]), οι οποίοι πρότειναν μία παραλλαγή της παραπάνω διαδικασίας, η οποία, σε πολλές περιπτώσεις, επιδεικνύει εφάμιλλα αποτελέσματα όσον αφορά στην ελαχιστοποίηση του εύρους ζώνης και της κατατομής, ενώ υπερτερεί καταφανώς όσον αφορά στη συνολική απαιτούμενη χρονική πολυπλοκότητα. Ο αλγόριθμος των Gibbs, et al, αποτελείται από τρεις επιμέρους αλγοριθμικές διαδικασίες, η συνοπτική περιγραφή των οποίων δίνεται στη συνέχεια:

ΑΛΓΟΡΙΘΜΟΣ 1: Ο αλγόριθμος αυτός βασίστηκε στην παρατήρηση ότι οι δομές επιπέδων που έχουν μεγάλο πλήθος επιπέδων, έχουν συνήθως μικρό εύρος. Από την παρατήρηση αυτή προέκυψε το συμπέρασμα ότί ένας συνδυασμός δύο τέτοιων δομών επιπέδων, οποίες να έχουν ως κορυφές ρίζες τα άκρα μιας *ψευδο-διαμέτρου*, θα ήταν ιδεώδης. Με βάση τα παραπάνω, ο πρώτος αλγόριθμος βρίσκει τα άκρα v και u μιας ψευδο-διαμέτρου και κατόπιν παράγει τις δομές επιπέδων $LS_v(G)$ και $LS_u(G)$ με ρίζες τα άκρα v και u, αντίστοιχα.

ΑΛΓΟΡΙΘΜΟΣ II : Ο συνδυασμός των δομών επιπέδων $LS_v(G)$ και $LS_u(G)$ αποδίδει μία νέα δομή επιπέδων, της οποίας το εύρος ζώνης είναι, συνήθως, μικρότερο σε σχέση με το εύρος ζώνης οποιασδήποτε από τις αρχικές δομές επιπέδων. Κατά συνέπεια, ο δεύτερος αλγόριθμος ελαχιστοποιεί το μέγιστο εύρος επιπέδου μέσω της κατασκευής μιας νέας δομής επιπέδων LS(G), η οποία προκύπτει από το συνδυασμό των δομών που παράχθηκαν από τον πρώτο αλγόριθμο.

ΑΛΓΟΡΙΘΜΟΣ III : Ο τρίτος αλγόριθμος εκτελεί μία διαδικασία αρίθμησης με την εκχώρηση διαδοχικών θετικών ακεραίων στις κορυφές του γραφήματος, ξεκινώντας από τις κορυφές του πρώτου επιπέδου της δομής που προέκυψε και καταλήγοντας σε αυτές του τελευταίου επιπέδου της. Η διαδικασία αρίθμησης που εφαρμόζεται είναι παρόμοια με αυτή της δεύτερης φάσης του αλγόριθμου RCM, αν και στην περίπτωση αυτή έχουν γίνει κάποιες τροποποιήσεις οι οποίες οφείλονται στο ότι η δομή δεδομένων που προκύπτει από τον αλγόριθμο II είναι πιο γενικής μορφής από τις αρχικές δομές.

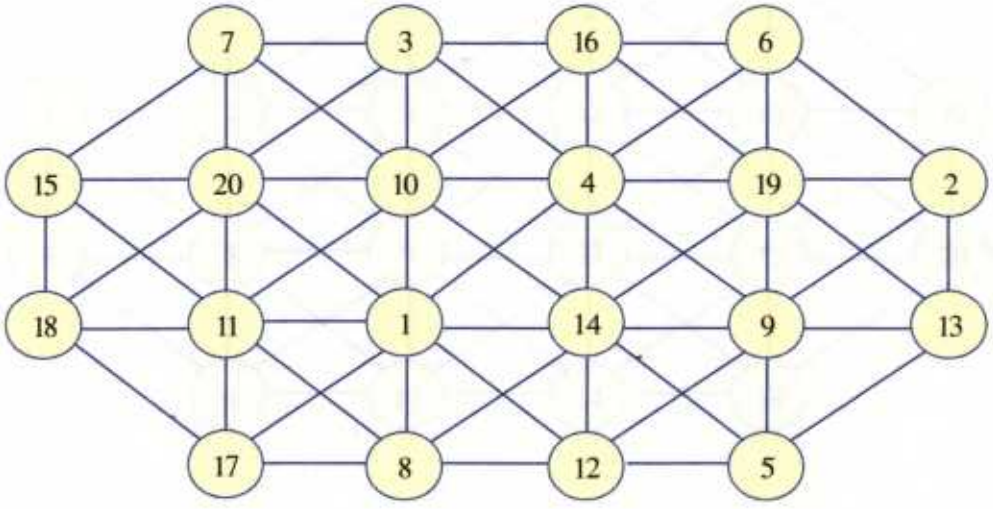
4.4 Διαγραμματική Απεικόνιση Σειριακών Αλγορίθμων

Στην παράγραφο αυτή θα παρουσιαστεί η εφαρμογή των δύο αλγόριθμων που περιγράφηκαν κάνοντας χρήση ενός συγκεκριμένου παραδείγματος. Αρχικά, κάνοντας χρήση των κατάλληλων γράφων, θα δοθεί η διαγραμματική απεικόνιση του αλγόριθμου των Cuthill-Mckee και στη συνέχεια η αντίστοιχη διαγραμματική απεικόνιση του αλγόριθμου των Gibbs, et al.

Έστω ότι ο συντελεστής πίνακας A έχει τη μορφή

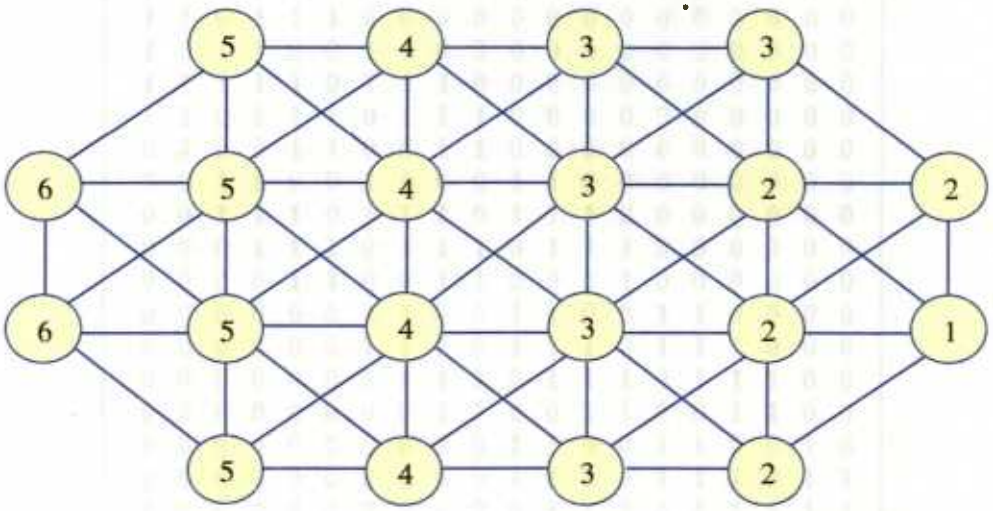
$$A = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{bmatrix} \qquad (4.4:1)$$

όπου το '1' υποδηλώνει τη θέση των μη μηδενικών στοιχείων. Το γράφημα με τις αριθμημένες κορυφές που σχετίζεται με τον προηγούμενο αραιό πίνακα περιγράφεται στη συνέχεια (4.4:2).



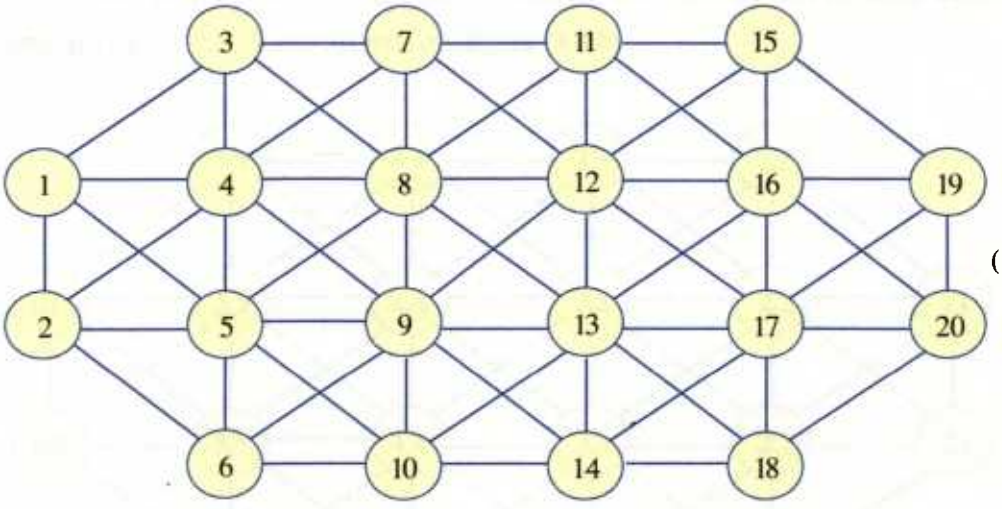
(4.4:2)

Επειδή ισχύει ότι LD=4, ο αλγόριθμος των Cuthill-Mckee δημιουργεί τις δομές επιπέδων με ρίζες τις κορυφές 6, 2, 13, 5, 17, 18, 15 και 7. Για παράδειγμα, η δομή επιπέδων με ρίζα την κορυφή 13 έχει την ακόλουθη μορφή:



(4.4:3)

Οι υπόλοιπες επτά δομές επιπέδων έχουν παρόμοια μορφή. Χρησιμοποιώντας τη δομή επιπέδων (4.4:3), προκύπτει η παρακάτω αντίστροφη αρίθμηση (4.4:4):

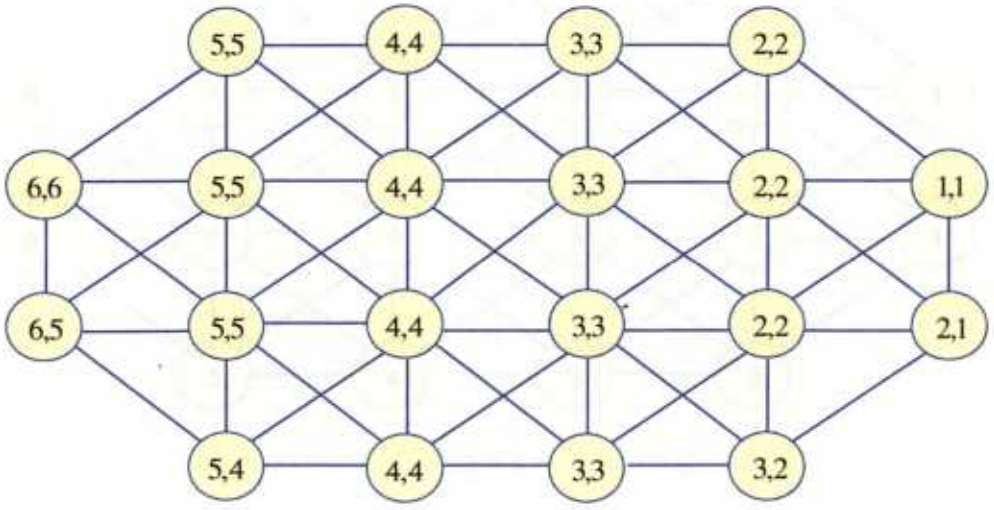


(4.4:4)

Ο συμπυκνωμένος αραιός πίνακας που προκύπτει και του οποίου το εύρος ζώνης και η κατατομή ισούνται με 5 και 79, αντίστοιχα, έχει τη μορφή (4.4:5):

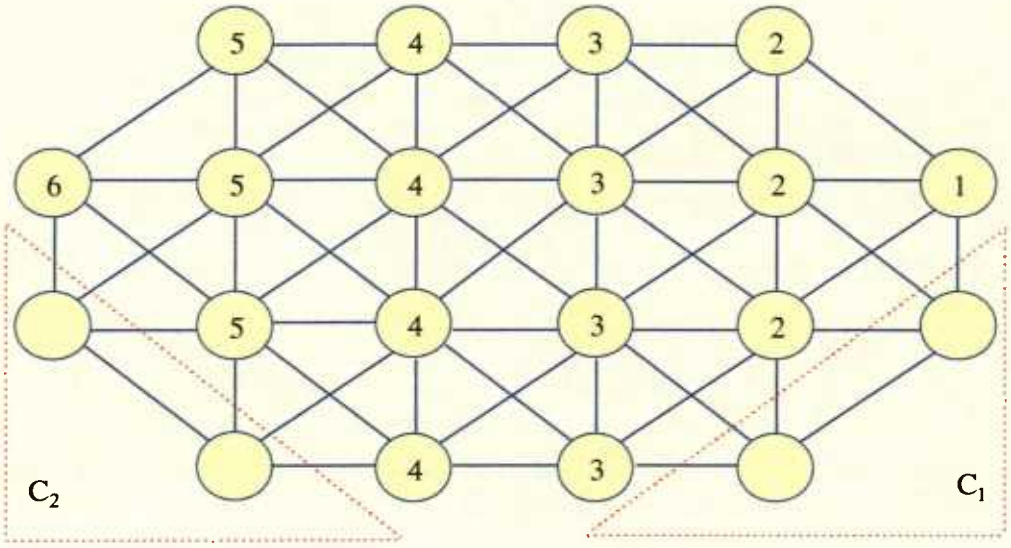
$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix} \qquad (4.4:5)$$

Όσον αφορά στον αλγόριθμο των Gibbs, et al, αρχικά δημιουργούνται οι δομές επιπέδων με ρίζες τις κορυφές v=2 και u=15. Τα *ζεύγη επιπέδων* (level pairs) που προκύπτουν με βάση αυτές τις δομές επιπέδων και σχετίζονται με κάθε κορυφή του γραφήματος (4.4:2), δίνονται στη συνέχεια (4.4:6).



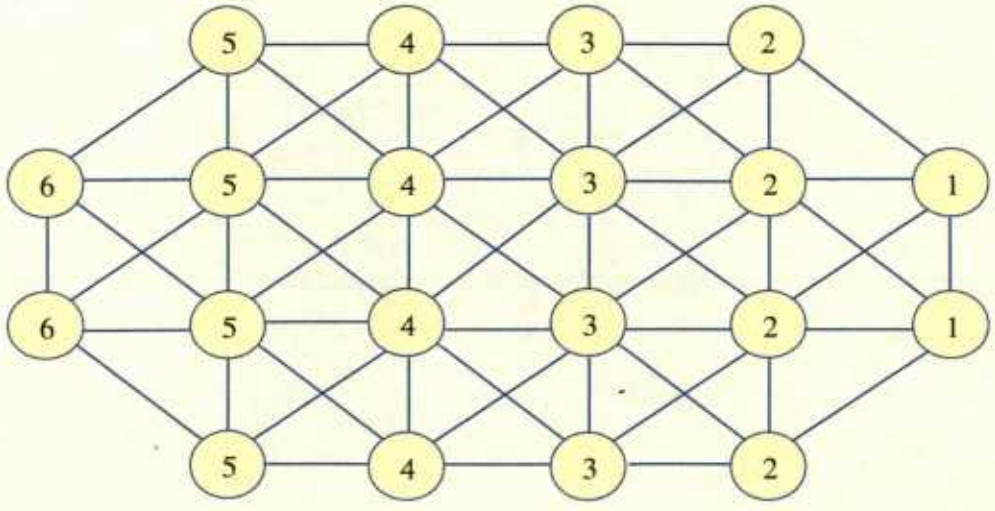
(4.4:6)

Κατά τη διάρκεια της δεύτερης φάσης του αλγόριθμου διενεργείται η εκχώρηση των κορυφών του γραφήματος που έχουν ζεύγος επιπέδων της μορφής (i, i), στο επίπεδο N_i. Οι ξεχωριστές συνεκτικές συνιστώσες του γραφήματος, που προκύπτουν μετά από μία τέτοια εκχώρηση περιγράφονται παρακάτω (4.4:7).



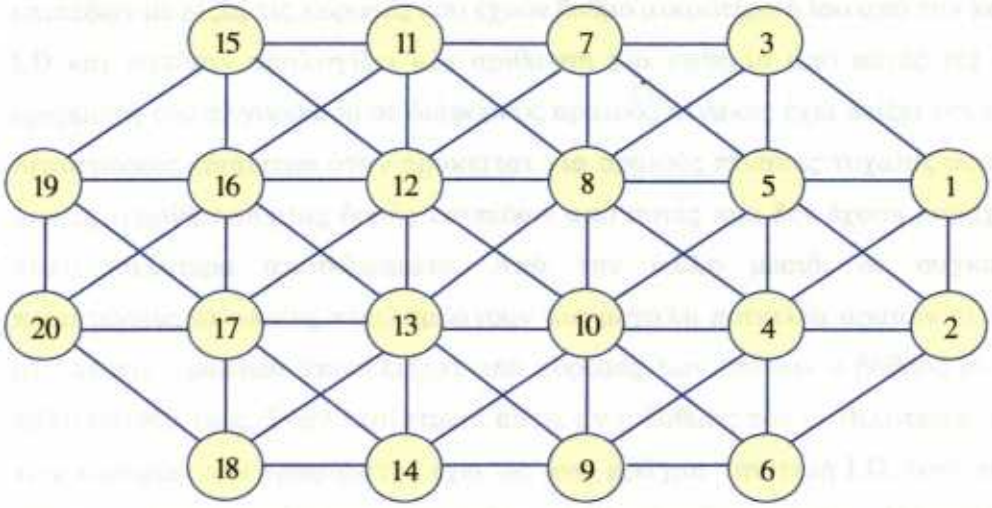
(4.4:7)

Μετά από την εκχώρηση των κορυφών των συνιστωσών C_1 και C_2 στα κατάλληλα επίπεδα προκύπτει η δομή επιπέδων (4.4:8)



(4.4:8)

Από την τελική αρίθμηση των κορυφών προκύπτει το γράφημα (4.4:9)



(4.4:9)

ενώ ο συμπυκνωμένος αραιός πίνακας που προκύπτει και του οποίου το εύρος ζώνης και η κατατομή ισούνται με 6 και 80, αντίστοιχα, δίνεται στη συνέχεια (4.4:10).

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
$$

(4.4:10)

4.5 Πολυεπιπεδική Συστολική Προσέγγιση

Όπως ήδη αναφέρθηκε, ο αλγόριθμος των Cuthill-Mckee δημιουργεί όλες τις δομές επιπέδων με ρίζες τις κορυφές που έχουν βαθμό μικρότερο ή ίσο από την παράμετρο LD και κατόπιν υπολογίζει μία αρίθμηση για καθεμία από αυτές τις δομές. Η εφαρμογή του αλγόριθμου σε διάφορους αραιούς πίνακες έχει δείξει ότι υπάρχουν περιπτώσεις, ιδιαίτερα όταν πρόκειται για αραιούς πίνακες τυχαίας μορφής, στις οποίες η αρίθμηση μιας δομής επιπέδων από αυτές που δεν έχουν παραχθεί δίνει πολύ καλύτερα αποτελέσματα. Από την άλλη μεριά, σε συγκεκριμένες περιπτώσεις, οι οποίες περιλαμβάνουν μία μεγάλη ποικιλία αραιών πινάκων, το αντίστοιχο γράφημα αποτελείται από κορυφές των οποίων ο βαθμός δε διαφέρει πολύ μεταξύ τους. Στην περίπτωση αυτή, αν ο βαθμός του μεγαλύτερου ποσοστού των κορυφών του γραφήματος έχει ως άνω φράγμα την τιμή LD, τότε παράγεται από τον αλγόριθμο ένα αρκετά μεγάλο πλήθος δομών επιπέδων. Όπως είδαμε στο προηγούμενο παράδειγμα, απαιτήθηκε η δημιουργία δομών επιπέδων με ρίζες *οκτώ* κορυφές οι οποίες αντιστοιχούσαν στο 40% του συνολικού πλήθους των κορυφών του γραφήματος. Επιπλέον, στην περίπτωση που όλες οι κορυφές του γραφήματος έχουν τον ίδιο βαθμό, τότε ο αλγόριθμος παράγει μία δομή επιπέδων για κάθε κορυφή του γραφήματος.

Ο αλγόριθμος των Gibbs, et al, ουσιαστικά, αποτέλεσε μία προσπάθεια βελτίωσης του αλγόριθμου RCM, η οποία αποδείχτηκε αρκετά ικανοποιητική όσον αφορά στη μείωση του εύρους ζώνης και της κατατομής που επιτυγχάνει. Πιο συγκεκριμένα, έχει αποδειχθεί ότι ο αλγόριθμος επιτυγχάνει μία μείωση του εύρους ζώνης και της κατατομής η οποία είναι συγκρίσιμη με αυτή του αλγόριθμου RCM, αλλά υπερτερεί σημαντικά όσον αφορά στον απαιτούμενο συνολικό χρόνο υλοποίησης. Ωστόσο, και σ' αυτή την περίπτωση υπάρχουν συνδυασμοί δομών επιπέδων οι οποίοι μολονότι δεν εξετάζονται μπορούν να παρέχουν καλύτερα αποτελέσματα.

Προκειμένου να προσδιοριστεί ο *πραγματικά βέλτιστος κόμβος εκκίνησης* δημιουργούνται όλες οι δυνατές δομές επιπέδων, υπολογίζονται όλες οι αντίστοιχες αριθμήσεις αυτών των δομών και κατόπιν επιλέγεται εκείνη που παράγει το μικρότερο εύρος ζώνης. Στην προκειμένη περίπτωση, το τίμημα για την εύρεση του βέλτιστου κόμβου εκκίνησης είναι η αύξηση της συνολικής χρονικής πολυπλοκότητας του αλγόριθμου. Με βάση, όμως, την ανάλυση που προηγήθηκε για τον αλγόριθμο RCM, σε πολλές περιπτώσεις παρατηρείται ότι η δημιουργία

των υπολοιπόμενων δομών επιπέδων, ο υπολογισμός των αριθμήσεών τους και, τέλος, ο υπολογισμός του εύρους ζώνης και της κατατομής που προκύπτουν από τις δομές αυτές, προκαλεί μία σχετικά μικρή επιβάρυνση όσον αφορά στη συνολική χρονική πολυπλοκότητα που απαιτείται.

4.5.1 Συστολική Μηχανή Πολλαπλών Λειτουργιών (Multiple Functional Engine, MFE)

Με δεδομένο το πρόβλημα της αύξησης της χρονικής πολυπλοκότητας προκειμένου για τον προσδιορισμό του βέλτιστου κόμβου εκκίνησης, η προσπάθεια επικεντρώθηκε, κυρίως, στην αξιοποίηση του ενυπάρχοντος παραλληλισμού του αλγόριθμου. Το πρώτο βήμα αφορούσε στη δημιουργία των δομών επιπέδων, μία κάθε φορά, κάνοντας χρήση συστολικών διατέξεων. Η βασική ιδέα που καθιστά εφικτή μία τέτοια υλοποίηση έγκειται στην αφιέρωση μιας ξεχωριστής *ομάδας* (block) συστολικών κελιών για την παραγωγή των κορυφών κάθε επιπέδου μιας δομής. Με άλλα λόγια, αν μία ομάδα συστολικών κελιών αντιστοιχεί στο επίπεδο i, τότε θεωρείται ότι παράγει τις κορυφές του επιπέδου i+1. Η συστολική διάταξη που χρησιμοποιείται λειτουργεί ως μία αρχιτεκτονική *ροής πληροφορίας* (dataflow architecture), δηλαδή η ενεργοποίηση ενός κελιού εξαρτάται από τη διαθεσιμότητα των απαιτούμενων δεδομένων και όχι από το χρονικό παλμό ενός ρολογιού.

Για την υλοποίηση της διαδικασίας αρίθμησης κάθε παραγόμενης δομής επιπέδων χρησιμοποιείται μία διαδικασία ταξινόμησης. Έτσι, τα αποτελέσματα κάθε ομάδας συστολικών κελιών (δηλαδή, κάθε επιπέδου) εισέρχονται ξεχωριστά σε μία άλλη γραμμική συστολική διάταξη, στην οποία πραγματοποιείται η ταξινόμηση των κορυφών κατά αύξουσα σειρά βαθμών. Το μέγεθος της συστολικής αυτής διάταξης προσδιορίζεται δυναμικά κάθε φορά, ανάλογα με το πλήθος των στοιχείων του συνόλου L_i που παράγονται κατά τη δημιουργία της δομής επιπέδων.

Τέλος, για τον υπολογισμό του εύρους ζώνης και της κατατομής χρησιμοποιείται ένα ξεχωριστό *σωληνωδίκτυο* (pipenet). Ένα σωληνωδίκτυο είναι, ουσιαστικά, μία δυναμικά διαρθρούμενη προγραμματιζόμενη συστολική διάταξη, η οποία μπορεί να χρησιμοποιηθεί για τον υπολογισμό διαφόρων σύνθετων διανυσματικών λειτουργιών (vector compound functions) (Hwang, et al, [44]). Η παρεχόμενη δυνατότητα προγραμματισμού βοηθάει στον ορισμό τόσο των

δυναμικών συνδέσεων, όσο και του πλήθους των καθυστερήσεων κατά μήκος ορισμένων διαδρομών διασύνδεσης. Πιο συγκεκριμένα, ένα σωληνωδίκτυο αποτελείται από πολλαπλούς λειτουργικούς αγωγούς (Functional Pipelines - FPs) οι οποίοι διασυνδέονται μέσω δύο διαραβδωτικών σωληνωδικτύων (Buffered Crossbar pipelined Networks - BCNs).

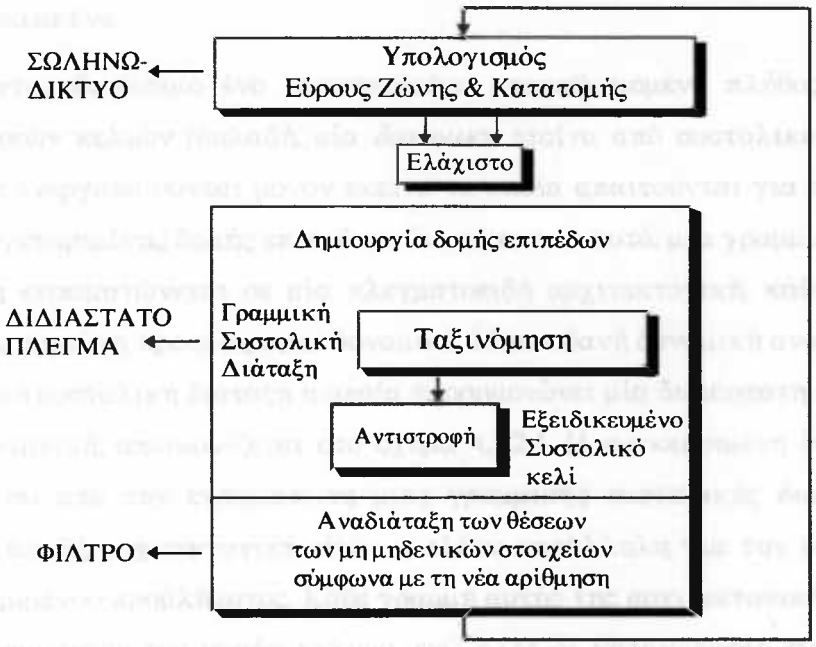
Από τα παραπάνω προκύπτει ότι, ένα σωληνωδίκτυο είναι μία αρχιτεκτονική στην οποία εφαρμόζεται σωλήνωση δύο επιπέδων. Το χαμηλότερο επίπεδο αντιστοιχεί στη σωλήνωση κατά μήκος των λειτουργικών αγωγών, ενώ το υψηλότερο επίπεδο αντιστοιχεί στη σωλήνωση μεταξύ των λειτουργικών αγωγών και διαμέσω των διαραβδωτικών σωληνωδικτύων. Το πλεονέκτημα των σωληνωδικτύων είναι ότι μπορούν να προσφέρουν περισσότερη ευελιξία σε σχέση με τις συστολικές διατάξεις, από την άποψη ότι πολλές ιδεατές συστολικές διατάξεις είναι δυνατό να προκύψουν από ένα και μόνο σωληνωδίκτυο. Οι συστολικές αυτές διατάξεις μπορούν να χρησιμοποιηθούν για την εκτέλεση διαφορετικών διανυσματικών λειτουργιών σε διαφορετικές χρονικές στιγμές και έτσι, ο υπολογισμός του εύρους ζώνης και της κατατομής μέσω ενός σωληνωδικτύου είναι εφικτός, εφόσον οι αντίστοιχοι υπολογισμοί μπορεί να θεωρηθεί ότι αποτελούν διανυσματικές λειτουργίες.

Περαιτέρω αξιοποίηση του ενυπάρχοντος παραλληλισμού επιτυγχάνεται στο βαθμό που όλες οι παραπάνω λειτουργίες πραγματοποιούνται με χρονική επικάλυψη. Αυτό σημαίνει ότι κατά τον υπολογισμό του εύρους ζώνης και της κατατομής από το σωληνωδίκτυο για την πρώτη αρίθμηση που έχει παραχθεί, λαμβάνει χώρα και η ταξινόμηση των κορυφών κάθε επιπέδου της δεύτερης δομής επιπέδων, στη δεύτερη συστολική διάταξη, ενώ ταυτόχρονα πραγματοποιείται και η δημιουργία της τρίτης δομής επιπέδων στην πρώτη συστολική διάταξη.

Όλες οι εξειδικευμένες αρχιτεκτονικές (special-purpose architectures), στις οποίες έγινε αναφορά στα προηγούμενα, περιγράφονται συνοπτικά και με αφηρημένη μορφή στο σχήμα 4.5.1-1. Κάθε τέτοια αρχιτεκτονική αναπαρίσταται από ένα *'μαύρο κουτί'* και συνδέεται με μερικές από τις υπόλοιπες με τέτοιον τρόπο ώστε ολόκληρη η διάρθρωση (configuration) να αποτελεί μία πλήρη μηχανή για τη λύση του προβλήματος της ελαχιστοποίησης του εύρους ζώνης και της κατατομής αραιών πινάκων. Η αναλυτική περιγραφή καθώς και η λειτουργία κάθε επιμέρους

συνιστώσας αυτής της μηχανής περιγράφονται στις επόμενες Παραγράφους του παρόντος Κεφαλαίου.

Η διαδικασία συμπύκνωσης ξεκινάει με τον υπολογισμό του εύρους ζώνης και της κατατομής του αρχικού αραιού πίνακα και συνεχίζεται με τη δημιουργία κάθε δομής επιπέδων και τον υπολογισμό της αντίστοιχης αρίθμησης. Η διαδικασία αυτή επαναλαμβάνεται περιοδικά, ενώ κάθε φορά επιλέγεται η αναδιάταξη εκείνη των κορυφών (δηλαδή, των αντίστοιχων γραμμών του αραιού πίνακα) η οποία αποδίδει το ελάχιστο, συγκριτικά, εύρος ζώνης.



Σχήμα 4.5.1-1: Συστολική Μηχανή Πολλαπλών Λειτουργιών

4.5.2 Αφαιρετική Διδιάστατη Πλεγματοειδής (2D-Mesh) Αρχιτεκτονική

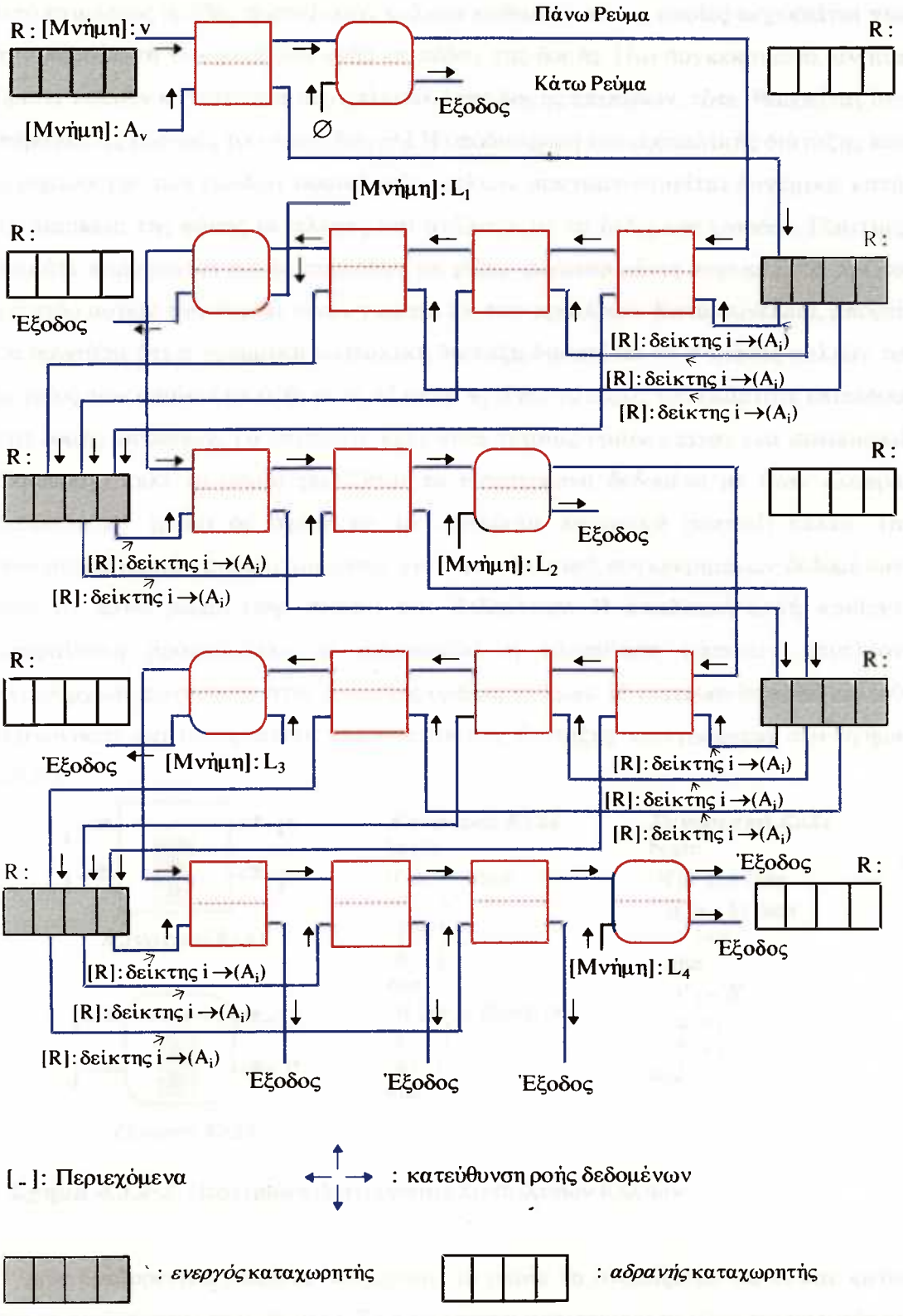
Όπως ήδη αναφέρθηκε, οι δομές επιπέδων είναι δυνατό να δημιουργηθούν κάνοντας χρήση συστολικών διατάξεων. Ο συστολικός υπολογισμός των δομών αυτών προυποθέτει μία προεπεξεργασία του αρχικού αραιού πίνακα η οποία συνεπάγεται την επεξεργασία του πίνακα, γραμμή προς γραμμή, έτσι ώστε να εντοπισθούν οι θέσεις των μη μηδενικών στοιχείων. Οι θέσεις αυτές, δηλαδή οι δείκτες των στηλών που περιέχουν μη μηδενικά στοιχεία, αποθηκεύονται ξεχωριστά για κάθε γραμμή δημιουργώντας έτσι μία δομή δεδομένων A_i, i=1,..., n, αντίστοιχα. Στις δομές δεδομένων A_i αποθηκεύονται όλοι οι δείκτες κάθε στήλης

που αντιστοιχούν σε μη μηδενικά στοιχεία της γραμμής i, εκτός από το δείκτη j για τον οποίο ισχύει ότι i=j.

Η δημιουργία μιας δομής επιπέδων, με ρίζα μία συγκεκριμένη κορυφή, απαιτεί το πολύ n+d συστολικά κελιά, όπου n είναι το μέγεθος του αραιού πίνακα και d είναι το *βάθος* (depth), δηλαδή το πλήθος των επιπέδων της εκάστοτε δομής. Επομένως, στη χειρότερη περίπτωση απαιτούνται 2n συστολικά κελιά. Η περίπτωση αυτή αναφέρεται στην εκφυλισμένη κατάσταση στην οποία το γράφημα που αντιστοιχεί στον υπό εξέταση αραιό πίνακα είναι μία λίστα με κάθε κορυφή να έχει βαθμό ίσο με δύο, εκτός από την πρώτη και την τελευταία κορυφή που έχουν βαθμό ίσο με ένα.

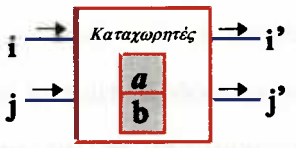
Έχοντας διαθέσιμο ένα συγκεκριμένο προκαθορισμένο πλήθος ισοδύναμων συστολικών κελιών (δηλαδή, μία *δυναμική* πισίνα από συστολικά κελιά), κάθε φορά θα ενεργοποιούνται μόνον εκείνα τα οποία απαιτούνται για τη δημιουργία μιας συγκεκριμένης δομής επιπέδων. Για το σκοπό αυτό, μία γραμμική συστολική διάταξη ενσωματώνεται σε μία πλεγματοειδή αρχιτεκτονική, κάθε γραμμή της οποίας μπορεί να προσδιοριστεί δυναμικά. Μία πιθανή δυναμική αναδιαρθρούμενη γραμμική συστολική διάταξη η οποία προσομοιώνει μία διδιάστατη πλεγματοειδή αρχιτεκτονική, απεικονίζεται στο σχήμα 4.5.2-1. Η συγκεκριμένη διάρθρωση που προκύπτει από την ενσωμάτωση μιας γραμμικής συστολικής διάταξης σε μία πλεγματοειδή αρχιτεκτονική είναι η πλέον κατάλληλη για την υλοποίηση του συγκεκριμένου προβλήματος. Κάθε γραμμή αυτής της αρχιτεκτονικής αντιστοιχεί σε μία συγκεκριμένη ομάδα κελιών, ενώ όλες οι κατακόρυφες *συνδέσεις* (links) μεταξύ των κελιών αγνοούνται, εκτός από αυτές που συνδέουν το οριακό κελί κάθε επιπέδου με το πρώτο κελί του επόμενου επιπέδου.

Η γραμμική συστολική διάταξη είναι μία αρχιτεκτονική ροής πληροφορίας με την έννοια ότι η λειτουργία της εξαρτάται από τη διαθεσιμότητα των εκάστοτε απαιτούμενων δεδομένων. Αυτό το *στοιχειοδηγούμενο* (data driven) σχήμα απαιτεί ειδικούς μηχανισμούς ανίχνευσης της διαθεσιμότητας των δεδομένων. Επιπλέον, η κατεύθυνση της ροής των δεδομένων εναλλάσσεται μεταξύ διαδοχικών ομάδων συστολικών κελιών. Με άλλα λόγια, ο προσανατολισμός των κελιών μιας ομάδας είναι προς την αντίθετη κατεύθυνση σε σχέση με αυτόν της προηγούμενης και της επόμενης ομάδας.

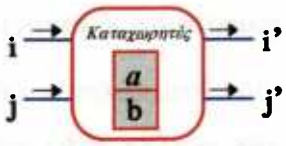


Σχήμα 4.5.2-1: Δυναμική Πλεγματοειδής Αρχιτεκτονική

Η συστολική αυτή διάταξη, όπως ήδη αναφέρθηκε, θεωρείται ότι αποτελείται από επιμέρους ομάδες συστολικών κελιών καθεμία από τις οποίες εκχωρείται για την παραγωγή των κορυφών ενός επιπέδου της δομής. Πιο συγκεκριμένα, αν μία ομάδα κελιών αντιστοιχεί στο επίπεδο *i* της δομής επιπέδων, τότε θεωρείται ότι παράγει τις κορυφές του επιπέδου *i+1*. Η υποδιαίρεση του συστολικής διάταξης και η δημιουργία των ομάδων συστολικών κελιών πραγματοποιείται δυναμικά κατά τη διάρκεια της φάσης εκτέλεσης και ανάλογα με τα δεδομένα εισόδου. Εξαιτίας του ότι παράγονται δομές επιπέδων με ρίζες συγκεκριμένες κορυφές, το πρώτο επίπεδο αυτών των δομών είναι γνωστό εκ των προτέρων. Κατά συνέπεια, μπορεί να θεωρηθεί ότι η γραμμική συστολική διάταξη διαιρείται σε *d* ομάδες κελιών το μέγεθος των οποίων ισούται με w_i+1, όπου w_i είναι το εύρος του εκάστοτε επιπέδου της δομής επιπέδων. Το επιπλέον κελί κάθε τέτοιας ομάδας είναι ένα *συνοριακό* (boundary) κελί το οποίο χειρίζεται τα εισερχόμενα δεδομένα με έναν ελαφρά διαφορετικό τρόπο σε σχέση με τα υπόλοιπα *κανονικά* (normal) κελιά. Τα συνοριακά κελιά χρησιμοποιούνται για την απαλοιφή συγκεκριμένων δεδομένων από το *πάνω ρεύμα* (top stream) των δεδομένων. Η απαλοιφή αυτή κρίθηκε απαραίτητη προκειμένου να αποφευχθεί η υλοποίηση κάποιων επιπλέον λειτουργιών σύγκρισης στις επόμενες ομάδες κελιών. Η λειτουργία κάθε κελιού (κανονικού και συνοριακού) της συστολικής διάταξης περιγράφεται στο σχήμα 4.5.2-2.



Κανονικό Κελί



Οριακό Κελί

Κανονικό Κελί

```
begin
if (a > b) then
  i' := b
  j' := b
  b := j
else
  if (a = b) then b := j
  i' := a
  a := i
end
```

Συνοριακό Κελί

```
begin
 if (a ≤ b) then
  if (a < b) then
   i' := a
  else
   i' := 'δ'
   a := i
   b := j
end
```

Σχήμα 4.5.2-2: Περιγραφή Λειτουργίας Συστολικών Κελιών

Δύο διαφορετικά ρεύματα δεδομένων, τα οποία θα ονομάζουμε *πάνω* και *κάτω* ρεύμα, εισέρχονται σε κάθε κελί. Το πάνω ρεύμα αντιστοιχεί σε όλες τις κορυφές οι οποίες έχουν ήδη εκχωρηθεί σε κάποιο επίπεδο της δομής επιπέδων, ενώ το κάτω

ρεύμα αντιστοιχεί σε όλες τις δομές δεδομένων A_i με τις οποίες το πάνω ρεύμα θα πρέπει να συγκριθεί προκειμένου να προσδιοριστούν οι κορυφές που δεν έχουν ακόμη εκχωρηθεί σε κάποιο επίπεδο.

Για τη δημιουργία μιας συγκεκριμένης δομής επιπέδων με ρίζα μία κορυφή, έστω v, θεωρούμε αρχικά ότι η κορυφή v εισέρχεται στον καταχωρητή ***a*** του πρώτου κελιού και ότι η δομή δεδομένων A_v εισέρχεται στον καταχωρητή ***b*** του ίδιου κελιού. Κάτι τέτοιο προυποθέτει ότι η έχει προηγηθεί η αρχικοποίηση της εκάστοτε δομής επιπέδων μέσω της εκχώρησης της κορυφής v στο πρώτο επίπεδο της δομής. Η κορυφή v ακολουθείται από ένα ειδικού τύπου σήμα εισόδου, που συμβολίζεται ως **EOD** (**E**nd **O**f **D**ata), το οποίο χρησιμοποιείται και στα δύο ρεύματα δεδομένων για να δηλώσει το τέλος των δεδομένων εισόδου και κατά συνέπεια, το τέλος της διάδοσης των δεδομένων.

Ένα ακόμη ειδικού τύπου σήμα το οποίο χρησιμοποιείται επίσης, συμβολίζεται με ***δ*** και εισάγει την έννοια των *ψευδο-στοιχείων* (pseudo-elements) τα οποία πιθανόν να εισέρχονται στη συστολική αρχιτεκτονική. Τα δύο αυτά σήματα υφίστανται ιδιαίτερη μεταχείριση όταν εισέρχονται σε κάποιο κελί. Έτσι, όταν ένα EOD σήμα εκχωρείται στον καταχωρητή ***a*** ενός κελιού, τότε τα περιεχόμενα του καταχωρητή ***b*** εξέρχονται και από τις δύο (πάνω και κάτω) γραμμές εξόδου του κελιού. Από την άλλη μεριά, όταν ένα EOD σήμα εκχωρείται στον καταχωρητή ***b*** ενός κελιού, τότε τα περιεχόμενα του καταχωρητή ***a*** εξέρχονται μόνο από από την πάνω γραμμή εξόδου του κελιού. Τέλος, στην περίπτωση που ένα ψευδο-στοιχείο ***δ***, εισέρχεται σε οποιονδήποτε από τους καταχωρητές ***a*** ή ***b*** ενός κελιού, τότε η λειτουργία του κελιού αυτού αναστέλλεται εφόσον τα δεδομένα που απαιτούνται δεν είναι ακόμη διαθέσιμα.

Τα αποτελέσματα κάθε ομάδας κελιών εξέρχονται μόνο από την κάτω γραμμή εξόδου κάθε κελιού και αποθηκεύονται στις κατάλληλες θέσεις μνήμης που αντιστοιχούν στα διακεκριμένα επίπεδα κορυφών L_i , i=1,...,d, της δεδομένης δομής επιπέδων. Κάθε αποτέλεσμα σχετίζεται με ένα δείκτη ο οποίος υποδηλώνει το συγκεκριμένο κελί της ομάδας από την οποία προέκυψε. Επιπλέον, τα αποτελέσματα αποθηκεύονται και στον κατάλληλο καταχωρητή της επόμενης ομάδας συστολικών κελιών, χωρίς τους αντίστοιχους δείκτες. Ακόμη και αν περισσότερα από ένα αποτελέσματα παράγονται από τα κελιά κατά την ίδια

χρονική περίοδο, δε δημιουργείται πρόβλημα όσον αφορά στην αποθήκευσή τους τόσο στη μνήμη, όσο και στον εκάστοτε καταχωρητή, εφόσον τα σήματα ελέγχονται σε διακριτά χρονικά διαστήματα. Αν όλα τα κελιά μιας ομάδας δεν έχουν να παράγουν άλλα αποτελέσματα τότε ένα EOD σήμα προστίθεται στον κατάλληλο καταχωρητή.

Το πλήθος των κελιών που πρόκειται να χρησιμοποιηθούν από μία ομάδα προσδιορίζεται από το πλήθος των κορυφών που έχουν αποθηκευτεί στον αντίστοιχο καταχωρητή. Κάθε στοιχείο ενός καταχωρητή είναι ένας δείκτης σε μία συγκεκριμένη δομή δεδομένων A_i. Οι καταχωρητές λειτουργούν υπό μορφή ουράς Πρώτο-μέσα Πρώτο-έξω (First In First Out - FIFO) και τα περιεχόμενά τους διαδίδονται, ένα κάθε φορά, στην κάτω γραμμή εισόδου ενός διαφορετικού συστολικού κελιού της ίδιας ομάδας. Η κατεύθυνση αυτής της διάδοσης εξαρτάται από τον προσανατολισμό της συγκεκριμένης ομάδας κελιών. Ουσιαστικά, το πάνω ρεύμα δεδομένων ρέει προς μία συγκεκριμένη κατεύθυνση καθόλη τη διάρκεια της διάδοσής του μέσα στη συστολική διάταξη, αλλά η ενσωμάτωση αυτής της διάταξης σε μία πλεγματοειδή αρχιτεκτονική δίνει την εντύπωση της εναλλαγής της κατεύθυνσης μεταξύ διαδοχικών ομάδων συστολικών κελιών.

Όταν ο εκάστοτε *ενεργός* καταχωρητής δεν έχει, προσωρινά, άλλα περιεχόμενα να διανείμει στα κελιά μιας ομάδας, ούτε καν ένα σήμα EOD, τότε υπονοείται η ύπαρξη ενός ψευδο-στοιχείου προκειμένου να ανασταλεί η λειτουργία ενός δεδομένου κελιού. Ως δεδομένα στην κάτω γραμμή εισόδου του συνοριακού συστολικού κελιού θεωρούνται τα στοιχεία του επιπέδου L_i της δομής επιπέδων που παράγεται τη δεδομένη χρονική στιγμή. Τα επίπεδα κορυφών που παράγονται ταξινομούνται στη συνέχεια κατά αύξουσα σειρά κορυφών. Έτσι, αν ένα συνοριακό κελί ανήκει στην ομάδα i, τότε το κάτω ρεύμα εισόδου αυτού του κελιού είναι το ταξινομημένο επίπεδο L_{i-1}. Όσον αφορά στο κάτω ρεύμα εισόδου του συνοριακού κελιού της πρώτης ομάδας, αυτό ισούται πάντοτε με το κενό σύνολο, $\varnothing$.

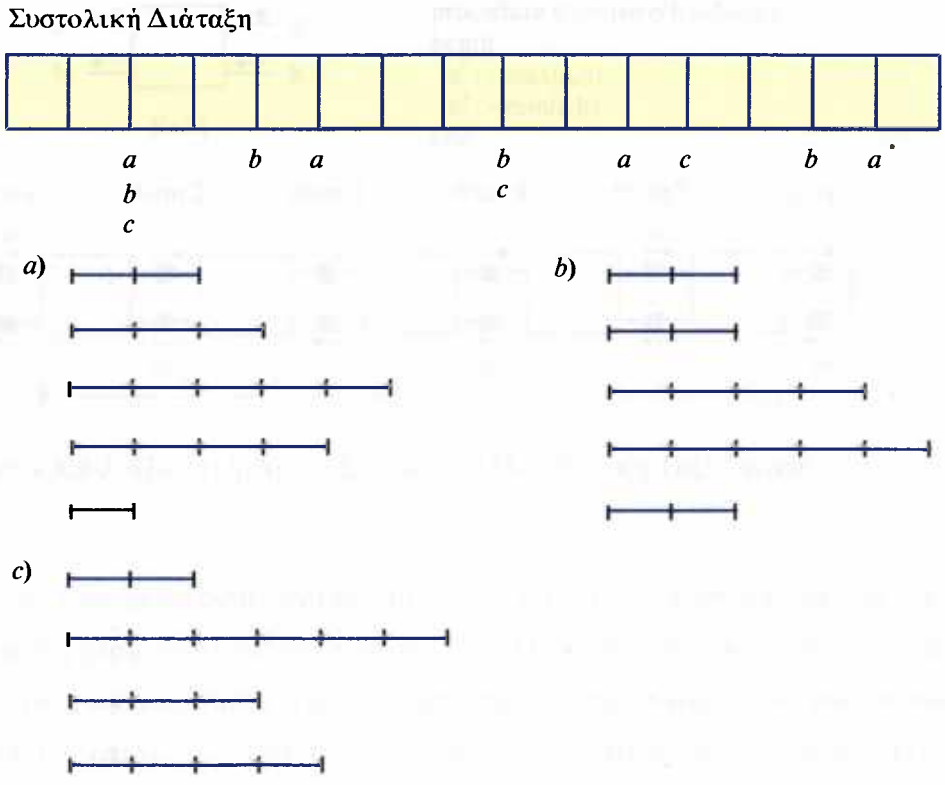
Σε σχέση με το πάνω ρεύμα δεδομένων, θα πρέπει να σημειωθεί ότι το μέγεθός του αυξομειώνεται δυναμικά καθώς ο συστολικός αλγόριθμος εξελίσσεται, εφόσον οι κορυφές που εκχωρούνται σε κάποιο επίπεδο συμπεριλαμβάνονται κάθε φορά στο ρεύμα αυτό και κατόπιν κάποιες από αυτές αφαιρούνται μέσω του συνοριακού συστολικού κελιού κάθε ομάδας. Γι' αυτό το λόγο κάποιες δυναμικά προσδιοριζόμενες *καθυστερήσεις* (delays) παρεμβάλονται μεταξύ των κελιών και

κατά μήκος του πάνω ρεύματος, προκειμένου να αποφευχθεί η επικάλυψη χρήσιμων δεδομένων. Σύμφωνα με τη συγκεκριμένη λειτουργία του συνοριακού κελιού δεν υπάρχει η ανάγκη για τέτοιου είδους καθυστερήσεις μεταξύ του συνοριακού κελιού και του αμέσως προηγούμενου κελιού της ίδιας ομάδας. Επιπλέον, όπως ήδη αναφέρθηκε, η απαλοιφή συγκεκριμένων δεδομένων από το πάνω ρεύμα είναι απαραίτητη. Όταν μία τέτοια απαλοιφή πραγματοποιείται από το συνοριακό κελί, τότε στέλνεται ένα ψευδο-στοιχείο στη θέση του στοιχείου που απαλείφθηκε.

Μετά από την παραγωγή του επιπέδου κορυφών *i* μιας συγκεκριμένης δομής επιπέδων, τα κελιά της ομάδας *i-1* γίνονται και πάλι διαθέσιμα. Κάτι τέτοιο συνεπάγεται ότι αυτά τα κελιά επιστρέφουν στην αρχική πισίνα από την οποία ελήφθησαν, προκειμένου να χρησιμοποιηθούν για τη δημιουργία της επόμενης δομής επιπέδων. Έχει αποδειχθεί ότι η δημιουργία της επόμενης δομής επιπέδων μπορεί να αρχίσει μετά από την αποδέσμευση των δύο πρώτων ομάδων κελιών της διάρθρωσης που χρησιμοποιείται για τη δημιουργία της αμέσως προηγούμενης δομής επιπέδων. Αυτό γίνεται για να αποφευχθεί η πιθανότητα επικάλυψης των περιεχομένων των καταχωρητών.

Κάθε συγκεκριμένη συστολική διάρθρωση που κατασκευάζεται για τη δημιουργία μιας δομής επιπέδων σχετίζεται με έναν *μετρητή* (counter). Τα περιεχόμενα του μετρητή αυτού αυξάνονται κατά ένα κάθε φορά που παράγεται ένα αποτέλεσμα, δηλαδή, κάθε φορά που μία κορυφή εκχωρείται σε κάποιο επίπεδο. Όταν η τιμή του μετρητή γίνει ίση με n, αυτό σημαίνει πως ολόκληρη η δομή επιπέδων έχει παραχθεί και έτσι δεν υπάρχει ανάγκη για περαιτέρω υπολογιστική δραστηριότητα στη συγκεκριμένη συστολική διάρθρωση. Έτσι, όλα τα υπόλοιπα κελιά τα οποία δεν έχουν ακόμη αποδεσμευτεί, γίνονται διαθέσιμα και επανεντάσσονται στην αρχική πισίνα των επεξεργαστών.

Η διαγραμματική απεικόνιση της δυναμικής αναδιάρθρωσης της συστολικής διάταξης για τη δημιουργία τριών διαδοχικών δομών επιπέδων δίνεται στο σχήμα 4.5.2-3. Το μέγεθος των διακεκριμένων ομάδων κελιών για κάθε διαφορετική συστολική διάρθρωση υποδηλώνεται από τους δείκτες *a*, *b*, και *c*, αντίστοιχα.

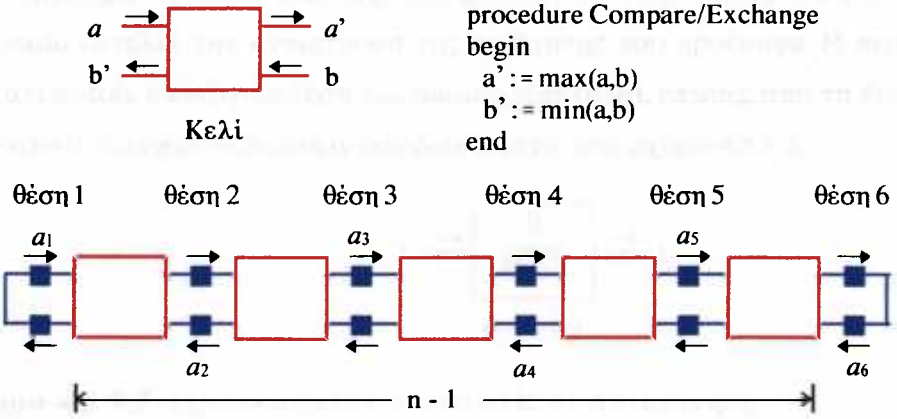


Σχήμα 4.5.2-3: Δυναμική Αναδιάρθρωση του Συστολικού Πίνακα

4.5.3 Ταξινόμηση σε Συστολική Γραμμική Διάταξη

Μετά από τη δημιουργία μιας δομής επιπέδων απαιτείται η διαδικασία επαναρίθμησης των κορυφών της δομής αυτής. Μία τέτοια διαδικασία συνεπάγεται την εκχώρηση διαδοχικών θετικών ακεραίων στις κορυφές της συγκεκριμένης δομής ξεκινώντας από το πρώτο επίπεδο και καταλήγοντας στο τελευταίο. Για την ταξινόμηση n στοιχείων, απαιτείται μία συστολική διάταξη μεγέθους *n-1*. Η μορφή μιας τέτοιας συστολικής διάταξης, η ροή των δεδομένων και η λειτουργία του χρησιμοποιούμενου συστολικού κελιού δίνεται στο σχήμα 4.5.3-1.

Ένα ζεύγος θέσεων αποθήκευσης τοποθετείται μεταξύ κάθε ζεύγους συστολικών κελιών. Οι δύο αυτές θέσεις χρησιμοποιούνται εναλλακτικά σε διαδοχικές χρονικές περιόδους για την αποθήκευση του αντίστοιχου στοιχείου της ακολουθίας. Κατά την έναρξη του αλγόριθμου οι αριθμοί που πρόκειται να ταξινομηθούν τοποθετούνται στις θέσεις αποθήκευσης. Σε κάθε διαφορετικό χρονικό βήμα ενεργοποιούνται μόνο εκείνα τα κελιά στα οποία εισέρχονται



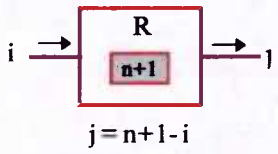
Σχήμα 4.5.3-1: Περιττή-Άρτια Συστολική Μεταθετική Ταξινόμηση

δεδομένα. Στη χειρότερη περίπτωση, απαιτούνται *n-1* χρονικά βήματα για την ταξινόμηση μιας ακολουθίας n στοιχείων. Η συγκεκριμένη συστολική διάταξη μπορεί να χρησιμοποιηθεί για την ταυτόχρονη ταξινόμηση δύο διακεκριμένων ακολουθιών στοιχείων. Για το σκοπό αυτό, η δεύτερη ακολουθία στοιχείων θα πρέπει να τοποθετηθεί στις υπόλοιπες ενδιάμεσες θέσεις αποθήκευσης πριν από την έναρξη του αλγόριθμου.

Με βάση την ανάλυση της συστολικής διαδικασίας για τη δημιουργία μιας δομής επιπέδων, καθώς και την περιγραφή της απαιτούμενης διαδικασίας επαναρίθμησης των κορυφών της δομής αυτής, προκύπτει ότι τρεις διαφορετικές διαδικασίες ταξινόμησης απαιτούνται για τις κορυφές κάθε επιπέδου μιας δομής. Καθεμία από αυτές τις διαδικασίες αντιστοιχεί στην ταξινόμηση κατά αύξουσα σειρά κορυφής, αύξουσα σειρά συστολικού κελιού (δηλαδή, του κελιού από το οποίο η εκάστοτε κορυφή έχει παραχθεί) και τέλος, ταξινόμηση κατά αύξουσα σειρά βαθμών κορυφής (για εκείνες τις κορυφές που έχουν προκύψει από το ίδιο κελί). Η δυνατότητα ταυτόχρονης ταξινόμησης δύο ακολουθιών στοιχείων, κατά την ίδια χρονική περίοδο, επιτρέπει την ταυτόχρονη υλοποίηση των δύο πρώτων ταξινομήσεων, γεγονός το οποίο μειώνει τη συνολική χρονική πολυπλοκότητα κατά *n-1* χρονικά βήματα.

Μετά από την ταξινόμηση των κορυφών ενός συγκεκριμένου επιπέδου L_i, τα χρησιμοποιηθέντα κελιά αποδεσμεύονται προκειμένου να χρησιμοποιηθούν για την ταξινόμηση της επόμενης ακολουθίας στοιχείων. Στα στοιχεία των ταξινομημένων επίπεδων αποδίδεται η κατάλληλη αρίθμηση, ενώ οι

επαναριθμημένες πλέον κορυφές εισέρχονται σε ένα εξειδικευμένο συστολικό κελί, το οποίο εκτελεί την αντιστροφή της αρίθμησης που προέκυψε. Η περιγραφή του κελιού αυτού, η ενεργοποίηση του οποίου εξαρτάται, επίσης, από τη διαθεσιμότητα των απαιτούμενων δεδομένων εισόδου, δίνεται στο σχήμα 4.5.3-2.

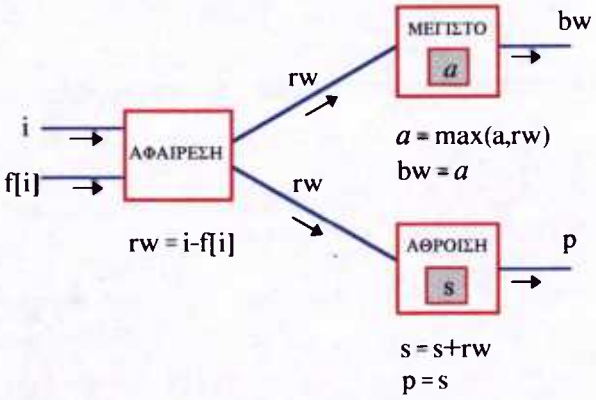


Σχήμα 4.5.3-2: Εξειδικευμένο Συστολικό Κελί Αντιστροφής

Μετά από την αντιστροφή της αρίθμησης που παράγεται, η αναδιάταξη των μη μηδενικών στοιχείων του αρχικού αραιού πίνακα μπορεί να γίνει μέσω ενός *φίλτρου* (filter). Αυτό το φίλτρο μπορεί να είναι είτε ένα κομμάτι κώδικα ή ένα ειδικό δίκτυο κατάλληλο για την πραγματοποίηση της συγκεκριμένης απεικόνισης.

4.5.4 Υπολογισμός του Εύρους Ζώνης και της Κατατομής

Έχοντας αναδιατάξει τα στοιχεία του αρχικού αραιού πίνακα με βάση την αρίθμηση που προέκυψε, η ενέργεια που απομένει να γίνει είναι ο υπολογισμός του εύρους ζώνης και της κατατομής του νέου συμπυκνωμένου πίνακα. Για το σκοπό αυτό μπορεί να χρησιμοποιηθεί ένα πολύ απλό σωληνωδίκτυο, η περιγραφή και η λειτουργία του οποίου δίνονται στο σχήμα 4.5.4-1.



Σχήμα 4.5.4-1: Υπολογισμός του Εύρους Ζώνης και της Κατατομής σε ένα Σωληνωδίκτυο

Μία προεπεξεργασία του αραιού πίνακα που προέκυψε θα πρέπει να προηγηθεί του υπολογισμού του εύρους ζώνης και της κατατομής του. Η διαδικασία αυτή προυποθέτει τη δημιουργία της δομής δεδομένων f[i], για i=1, ..., n, η οποία θα περιέχει τους δείκτες της στήλης του αριστερότερου μη μηδενικού στοιχείου κάθε γραμμής.

Τέλος, μετά από τον υπολογισμό του εύρους ζώνης και της κατατομής του συμπυκνωμένου αραιού πίνακα που προέκυψε με βάση κάποια συγκεκριμένη αρίθμηση κορυφών, οι τιμές αυτών των παραμέτρων συγκρίνονται με τις αντίστοιχες τιμές που προέκυψαν από άλλες αριθμήσεις και επιλέγεται τελικά η αρίθμηση που παράγει τις μικρότερες τιμές όσον αφορά στο εύρος ζώνης και στην κατατομή του αρχικού αραιού πίνακα. Κάτι τέτοιο επιτυγχάνεται μέσω της χρήσης ενός εξειδικευμένου κελιού στο οποίο τα αποτελέσματα συγκρίνονται ανά ζεύγη και κάθε φορά επιλέγεται εκείνο με τη μικρότερη τιμή.

Κεφάλαιο 5

Συμβολικό Περιγραφικό Εργαλείο Απόδοσης Συστολικών Αλγορίθμων

5.1 Εισαγωγή

Τα συστήματα υπολογιστών *ειδικού σκοπού* χρησιμοποιούνται για την εκτέλεση εργασιών με εκτεταμένο υπολογιστικό φόρτο (computationally intensive), η διεκπεραίωση των οποίων απαιτείται σε πολλές επιστημονικές περιοχές. Οι αλγόριθμοι οι οποίοι έχουν κανονική δομή, ενώ ταυτόχρονα επιδεικνύουν έναν υψηλού επιπέδου ενυπάρχοντα παραλληλισμό απεικονίζονται άμεσα σε δίκτυα επεξεργαστών (PEs), όπως είναι οι συστολικές αρχιτεκτονικές. Τα αλγοριθμικά δομημένα υπολογιστικά δίκτυα καθρεφτίζουν την κατάλληλη ροή πληροφορίας, ενώ ενδείκνυνται για υλοποίηση με Πολύ Μεγάλης Κλίμακας Ολοκλήρωση (Very Large Scale Integration, VLSI) εξαιτίας της κανονικής τους δομής και του άμεσου τρόπου διασύνδεσης των επεξεργαστών τους. Κατά συνέπεια τα VLSI δίκτυα επεξεργαστών, εξαιτίας του ότι έχουν σχεδιαστεί με αυτόν τον τρόπο, θα πρέπει να χειρίζονται τη ροή των δεδομένων έτσι ώστε να διατηρούν και να χρησιμοποιούν αυτό το επίπεδο ολοκλήρωσης. Η αύξηση της υπολογιστικής πολυπλοκότητας των PEs οδηγεί, αναμφίβολα, στη ραγδαία μείωση του συνολικού χρόνου εκτέλεσης με το τίμημα, όμως, μιας σχετικά πολύπλοκης υπολογιστικής δραστηριότητας. Η δυνατότητα παρουσίασης αυτής της, πιθανώς, πολύπλοκης υπολογιστικής

δραστηριότητας προυποθέτει τη χρήση ενός *Συμβολικού Περιγραφικού Εργαλείου* (Symbolic Descriptive Tool, SDT) για τη μονόδρομη (unidirectional) αναπαράσταση στο χώρο, αποδοτικών συστολικών υπολογισμών, το οποίο θα υποστηρίζει και μία τεχνική προγραμματιστικής αναπαράστασης προκειμένου για τη διαμόρφωση των αντίστοιχων συστολικών προγραμμάτων.

Όπως ήδη αναφέρθηκε, η έννοια της συστολικής επεξεργασίας προτάθηκε αρχικά από τον Kung στις αρχές της δεκαετίας του '90. Από τότε και μέχρι σήμερα, διάφορες παραλλαγές των συστολικών διατάξεων έχουν προταθεί και σχεδιαστεί στο πλαίσιο πανεπιστημιακών και βιομηχανικών προγραμμάτων έρευνας και ανάπτυξης. Οι περισσότερες από τις πρώτες συστολικές σχεδιάσεις που προτάθηκαν βασίζονταν, κατά κύριο λόγο, σε ευρετικές προσεγγίσεις, υπό την έννοια ότι δεν ακολουθείται κάποια συστηματική διαδικασία αναπαράστασης των συστολικών υπολογισμών. Σήμερα, ένα από τα πλέον ενδιαφέροντα προβλήματα που σχετίζεται με τη συστολική επεξεργασία αφορά στην ανάπτυξη μιας μεθοδολογίας για την απεικόνιση ενός αλγόριθμου σε μία συστολική αρχιτεκτονική. Οι περισσότερες από τις μεθοδολογίες αναπαράστασης που αναπτύχθηκαν την τελευταία δεκαετία βασίζονται στην έννοια των διανυσμάτων εξάρτησης (dependence vectors) για τη διάταξη ως προς το χώρο και το χρόνο των σημείων (index points) που αναπαριστούν τον εκάστοτε αλγόριθμο.

Μέσω του εργαλείου SDT εφαρμόζεται μία συστηματική μεθοδολογία αναπαράστασης, τόσο διαγραμματικής, όσο και προγραμματιστικής, συστολικών αλγορίθμων που υλοποιούνται στις κατάλληλες, κάθε φορά, συστολικές αρχιτεκτονικές, οι οποίες διαθέτουν ένα βέλτιστο πλήθος PEs με πιθανώς, ελαφρά αυξημένη πολυπλοκότητα. Το εργαλείο SDT βασίζεται σε μία προσέγγιση που αφορά στην εξάρτηση των δεδομένων, ενώ ταυτόχρονα κάνει χρήση βασικών εννοιών της Προβολικής Γεωμετρίας. Από την άλλη μεριά, η χρήση αυτού του εργαλείου είναι δυνατό να οδηγήσει στο σχεδιασμό βέλτιστων επίπεδων (planar) συστολικών διατάξεων, τόσο ως προς το πλήθος των PEs, όσο και ως προς τον απαιτούμενο συνολικό χρόνο εκτέλεσης που συνεπάγεται η χρήση τους για την επίλυση ενός συγκεκριμένου προβλήματος.

Το εργαλείο SDT επιτρέπει μία διαγραμματική αναπαράσταση στο χώρο οποιουδήποτε συστολικού αλγόριθμου, η οποία προυποθέτει την τοποθέτηση των δεδομένων εισόδου, όπως επίσης και των PEs και των αποτελεσμάτων εξόδου σε

συγκεκριμένους υποχώρους του πρώτου οκτάεδρου (octant) του Z^3. Η διάταξη των υπολογισμών περιγράφεται μέσω ενός γραφήματος εξάρτησης, το οποίο αναλύεται περαιτέρω από την αντίστοιχη προγραμματιστική αναπαράσταση στο χώρο. Στη συνέχεια, και έχοντας προσδιορίσει τις *κατευθύνσεις προβολής* (directions of projection, $d\vec{0}p_s$), για τις οποίες είναι εφικτή η αναπαράσταση των υπολογισμών, ακολουθεί ο προσδιορισμός της αντίστοιχης σχέσης για την απεικόνιση της σωστής ροής των δεδομένων και των σωστών θέσεων των PEs σε κάθε προβολικό επίπεδο (projective plane). Τέλος, δίνεται μία προγραμματιστική αναπαράσταση, του εκάστοτε συστολικού αλγόριθμου που προκύπτει, στο χώρο των δύο διαστάσεων, κάνοντας χρήση εντολών πολλαπλής εκχώρησης (multiple assignment statements) προκειμένου να αποδοθεί η ενυπάρχουσα έννοια του συγχρονισμού.

5.2 Συμβολικό Περιγραφικό Εργαλείο Απόδοσης Συστολικών Υπολογισμών

Όπως ήδη αναφέρθηκε, οι περισσότερες συστολικές αρχιτεκτονικές σχεδιάσεις, στις οποίες υλοποιούνται οι συστολικοί υπολογισμοί παρουσιάζονται με έναν εντελώς αυθαίρετο και, συχνά, ασαφή τρόπο. Η αναπαράσταση των συστολικών αλγορίθμων στον τρισδιάστατο χώρο (3D-space) παρέχει μία καθολικά δυναμική άποψη της συστολικής υπολογιστικής δραστηριότητας, ενώ εκμεταλλεύεται τη δυνατότητα κατασκευής βέλτιστων και επίπεδων συστολικών σχεδιάσεων. Μία τέτοια αναπαράσταση επιτυγχάνεται μέσω της διαμόρφωσης ενός τρισδιάστατου υπολογιστικού χώρου στον οποίο, τόσο οι θέσεις των δεδομένων εισόδου και των PEs, όσο και η ροή των δεδομένων και οι κατευθύνσεις της υπολογιστικής δραστηριότητας προσδιορίζονται με βάση τις αναδρομικές εξισώσεις (reccurrence equations) που χρησιμοποιεί ο προς επίλυση αλγόριθμος (Sedukhin, [65]).

Ας συμβολίσουμε με Z^3 το σύνολο των σημείων του τρισδιάστατου Καρτεσιανού χώρου *(i,j,k)*, στον οποίο τα διανύσματα βάσης είναι τα $\vec{i}(1,0,0)$, $\vec{j}(0,1,0)$ και $\vec{k}(0,0,1)$, αντίστοιχα. Κάθε σημείο $p \in Z^3$ προσδιορίζεται μοναδικά από το διάνυσμα θέσης του. Έτσι, όταν γίνεται αναφερά σε ένα συγκεκριμένο σημείο αυτό θα συμβολίζεται με *p(i,j,k)*, ενώ το αντίστοιχο διάνυσμα θέσης θα συμβολίζεται με $\vec{p}(i,j,k)$. Ένας υποχώρος $P \subset Z^3$, στον οποίο αναπαρίσταται η υλοποίηση ενός συγκεκριμένου

αλγόριθμου, θεωρείται ως η ένωση τριών υποχώρων, οι οποίοι συνιστούν το χώρο του αρχικού ρεύματος των δεδομένων P_{in}, το χώρο των εσωτερικών υπολογισμών P_{int} και το χώρο των αποτελεσμάτων εξόδου P_{out}. Οι υποχώροι P_{in} και P_{out} τοποθετούνται στα σύνορα (borders) του υποχώρου P_{int}.

Η παρουσίαση των υπολογισμών στο χώρο, αναμφίβολα, απλοποιεί οποιαδήποτε υπολογιστική δραστηριότητα, με την έννοια ότι εξαλείφει την πολλαπλότητα των κατευθύνσεων που παρουσιάζει η ροή των δεδομένων, στις διδιάστατες συστολικές διατάξεις, παρέχοντας μία δυνατότητα απεικόνισης της ροής αυτής προς μία κατεύθυνση, αλλά σε πολλαπλά επίπεδα. Με τον τρόπο αυτό, οι συστολικοί υπολογισμοί που παρουσιάζουν μία πολυπλοκότητα στο διδιάστατο χώρο μπορούν εύκολα να απεικονιστούν και να γίνουν κατανοητοί.

Η απεικόνιση ενός συστολικού αλγόριθμου σε μία συστολική αρχιτεκτονική προυποθέτει τη χρήση *διανυσμάτων εξάρτησης* προκειμένου για τη διάταξη στο *χώρο* και στο *χρόνο* των σημείων στα οποία λαμβάνει χώρα κάποια υπολογιστική δραστηριότητα, συμπεριλαμβανομένων και των σημείων στα οποία εκχωρείται το αρχικό ρεύμα δεδομένων εισόδου. Τα διατεταγμένα αυτά σημεία αναπαρίστανται με *κόμβους* (nodes) σε ένα γράφημα εξαρτήσεων Γ_g το οποίο διαθέτει καθολικές εξαρτήσεις, και το οποίο, τελικά, οδηγεί σε ένα κατευθυνόμενο γράφημα Γ_ℓ με τοπικές εξαρτήσεις. Η πραγματική δομή των επεξεργαστών της συστολικής διάταξης μπορεί να προκύψει μέσω της προβολής του γραφήματος Γ_ℓ σε ένα χώρο επεξεργαστών μικρότερης διάστασης. Στο βαθμό που υπάρχουν περισσότερες της μιας επιτρεπτές κατευθύνσεις προβολής, τότε είναι δυνατόν να προκύψουν διαφορετικές συστολικές σχεδιάσεις.

Πιο συγκεκριμένα, η διάταξη της υπολογιστικής δραστηριότητας διάδοσης (computational activity propagation) ενός συστολικού αλγόριθμου στο χώρο P μπορεί να περιγραφεί από ένα γράφημα $\Gamma_g = (P, E_g)$, $E_g = \{\vec{Q}_i^3\}$, για i =1, ..., k, όπου το k παίρνει μία πεπερασμένη ακέραια τιμή. Τα διανύσματα $\vec{Q}_1^3, \vec{Q}_2^3, ..., \vec{Q}_k^3$ αποτελούν τα καθολικά διανύσματα εξάρτησης για τα δεδομένα εισόδου και την κατεύθυνση των υπολογισμών. Αυτά τα διανύσματα εξάρτησης υποδηλώνουν, επίσης, τις διασυνδέσεις (interconnections) μεταξύ των PEs εκφράζοντας τη συσχέτιση που υπάρχει ανάμεσα στον υπολογισμό που πρόκειται να πραγματοποιηθεί στο σημείο $p(i,j,k) \in P$ και στα αντίστοιχα δεδομένα εισόδου σε αυτό το σημείο επεξεργασίας.

Στη συνέχεια, το γράφημα εξαρτήσεων Γ_g αντικαθίσταται από ένα ισοδύναμο γράφημα $\Gamma_\ell = (P, E_\ell)$, $E_\ell = \{\vec{e}_i^{\,3}\}$, για i=1, ..., k, στο οποίο τα διανύσματα $\vec{e}_1^{\,3}, \vec{e}_2^{\,3}, \ldots, \vec{e}_k^{\,3}$ είναι τα σταθερά τοπικά διανύσματα εξάρτησης. Μερικά από αυτά τα διανύσματα ισούνται, συνήθως, με τα διανύσματα βάσης των αξόνων στο χώρο.

Η απαλοιφή των καθολικών εξαρτήσεων καθιστά εφικτή την απεικόνιση του γραφήματος Γ_ℓ πάνω σε μία συστολική διάταξη, κάνοντας χρήση μιας επιτρεπτής κατεύθυνσης προβολής $\pi_\rho(\pi_i, \pi_j, \pi_k)$.

Προκειμένου να προκύψει η συστολική διάταξη επιβάλλεται η χρήση ενός *πίνακα μετασχηματισμού* (transformation matrix), ο οποίος απεικονίζει ένα σημείο *p(i,j,k)* ∈ P πάνω στο προβολικό επίπεδο. Για μία συγκεκριμένη dθρ π_ρ, ο πίνακας μετασχηματισμού $L(\pi_\rho)$ που επιλέγεται, απεικονίζει το γράφημα $\Gamma_\ell = (P, E_\ell)$ στο αντίστοιχο προβολικό επίπεδο, ενώ ταυτόχρονα παρέχει την ακριβή διάταξη των PEs διατηρώντας την τοπικότητα στη διασύνδεσή τους. Επιπλέον, η ακριβής διάταξη των δεδομένων εισόδου στο προβολικό επίπεδο αποκτάται μέσω της απεικόνισης σε αυτό του χώρου P_{in}. Για μία συγκεκριμένη dθρ π_ρ, ο πίνακας μετασχηματισμού $L(\pi_\rho)$:

$$L(\pi_\rho) = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \end{bmatrix}$$

δεν προσδιορίζεται μοναδικά. Ωστόσο, τα στοιχεία όλων των πινάκων αυτού του τύπου θα πρέπει να πληρούν τις ακόλουθες συνθήκες:

- $\alpha_i, \beta_i \in \{-1, 0, 1\}$, i = 1,2,3. Η συνθήκη αυτή εξασφαλίζει την ύπαρξη ενός σταθερού ρυθμού διάδοσης των δεδομένων εισόδου στο προβολικό επίπεδο, ενώ το γεγονός ότι τα στοιχεία του πίνακα παίρνουν τιμές από το σύνολο {-1, 0, 1}, ικανοποιεί την ιδιότητα της τοπικότητας. Παρατηρούμε ότι, οι στήλες του πίνακα $L(\pi_\rho)$ αναπαριστούν τις κατευθύνσεις ροής των δεδομένων στο προβολικό επίπεδο, δηλαδή,

$$\vec{e}_1^{\,2} = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}, \qquad \vec{e}_2^{\,2} = \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix}, \qquad \vec{e}_3^{\,2} = \begin{bmatrix} \alpha_3 \\ \beta_3 \end{bmatrix}.$$

- $L(\pi_\rho)\pi_\rho = 0$.

- $(\alpha_1 \ \alpha_2 \ \alpha_3) \neq d*(\beta_1 \ \beta_2 \ \beta_3)$, $d \neq 0$. Η ιδιότητα αυτή υπονοεί ότι οι γραμμές του πίνακα είναι γραμμικά ανεξάρτητες.
- $|\alpha_1| + |\alpha_2| + |\alpha_3| > 0$, $|\beta_1| + |\beta_2| + |\beta_3| > 0$. Οι ανισότητες αυτές εξασφαλίζουν τη μοναδικότητα του προβολικού επιπέδου, εφόσον τουλάχιστον ένα από τα α_i's και β_i's είναι $\neq 0$.

Μετά από την επιλογή του κατάλληλου πίνακα μετασχηματισμού προσδιορίζεται ο *πραγματικός* χώρος εσωτερικών υπολογισμών $P_{int} \subset P$, ο οποίος αποτελείται από ένα σύνολο ακέραιων σημείων από το πρώτο οκτάεδρο του Z^3. Ακολουθεί ο προσδιορισμός του πλήθους των επιτρεπτών κατευθύνσεων προβολής για τον εκάστοτε συγκεκριμένο αλγόριθμο και, τελικά, ο πεπερασμένος χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ λαμβάνεται από τον P_{int} μέσω της απεικόνισης $P_{int} \ni p(i,j,k) \rightarrow p(u,v,w) \in P_{int}(\pi_\rho)$, στην οποία τα u, v, w ορίζονται με βάση τις παραμετρικές εξισώσεις ευθειών για ένα κατάλληλο, κάθε φορά, σύνολο σημείων στο χώρο και μία δεδομένη και επιτρεπτή κατεύθυνση προβολής.

Κάθε κόμβος $p(v,u,w)$ θεωρείται ότι αντιστοιχεί σε ένα PE, το οποίο υλοποιεί την κατάλληλη πράξη σύμφωνα με την περιγραφή της λειτουργίας του. Κατά συνέπεια, προκύπτει μία τρισδιάστατη συστολική διάταξη, η απεικόνιση της οποίας στο προβολικό επίπεδο δίνει μία συστολική διάταξη μικρότερης διάστασης. Στην τελευταία αυτή συστολική διάταξη, η θέση κάθε PE περιγράφεται από τις καρτεσιανές του συντεταγμένες που προσδιορίζονται από την ακόλουθη σχέση

$$\begin{bmatrix} x \\ y \end{bmatrix} = L(\pi_\rho) p(u,v,w)$$

Αντίστοιχα, θα πρέπει να προσδιοριστούν και οι θέσεις των δεδομένων εισόδου στο προβολικό επίπεδο. Κάτι τέτοιο προυποθέτει την αντιστοίχιση των θέσεων του χώρου P_{in} με αυτές του χώρου Z^3 χρησιμοποιώντας τη *συνάρτηση χρόνου* (timing function) $t(p_\varepsilon) = u+v+w+\beta$, όπου το $\varepsilon \in \{$ *είσοδοι δεδομένων* $\varepsilon_i \}$ και το β είναι μία σταθερά η οποία θα πρέπει να προσδιοριστεί. Η σταθερά αυτή αποτιμάται με βάση τη συνθήκη $t(p_{start}) = 0$, όπου το $p_{start} \in P_{int}(\pi_\rho)$ είναι το σημείο στο οποίο πρόκειται να εκτελεστεί ο πρώτος υπολογισμός. Συνεπώς, η σχέση $t(p_\varepsilon)$, προσδιορίζει τη χρονική στιγμή στην οποία λαμβάνει χώρα κάποιος υπολογισμός στο αντίστοιχο σημείο $p(u,v,w) \in P_{int}(\pi_\rho)$, θεωρώντας ότι κάθε υπολογισμός απαιτεί μία χρονική

μονάδα. Οι θέσεις $\tilde{p}_{\varepsilon}$, των δεδομένων εισόδου στο Z^3, που προκύπτουν τελικά, χρησιμοποιούνται προκειμένου να προσδιοριστούν οι αντίστοιχες θέσεις στο προβολικό επίπεδο μέσω της ακόλουθης σχέσης :

$$\begin{bmatrix} x \\ y \end{bmatrix}_{\varepsilon} = L(\pi_p)\ \tilde{p}_{\varepsilon}\ (u,v,w)$$

5.3 Προγραμματιστική Αναπαράσταση Συστολικών Αλγορίθμων

Οι συστολικοί αλγόριθμοι συνήθως αναπαρίστανται μέσω διαγραμμάτων που περιγράφουν ένα διατεταγμένο σύνολο συστολικών κελιών και τις μεταξύ τους διασυνδέσεις, καθώς και το είδος της επεξεργασίας που λαμβάνει χώρα σε κάθε κελί και την κίνηση των δεδομένων μεταξύ αυτών. Αν και αυτού του είδους η απεικόνιση παρουσιάζει μία αμεσότητα στην κατανόηση, ωστόσο η προγραμματιστική αναπαράσταση (Chandy, [24], Bekakos, [18]) των συστολικών αλγορίθμων επιτρέπει μία πιο συνοπτική παρουσίαση της συστολικής επεξεργασίας και παρέχει την άμεση δυνατότητα προσομοίωσής τους, στο βαθμό που κάτι τέτοιο είναι επιθυμητό. Προκειμένου να εκφραστεί ένας συστολικός αλγόριθμος, μέσω ενός προγράμματος, χρησιμοποιούνται παραδοσιακές προγραμματιστικές τεχνικές οι οποίες επαυξάνονται με κάποιες έννοιες απαραίτητες για την έκφραση του συγχρονισμού (synchronization) που λαμβάνει χώρα στο συστολικό δίκτυο. Η πιο σημαντική από αυτές είναι η έννοια της *εντολής πολλαπλής εκχώρησης* (multiple assignement statement), η οποία μπορεί να θεωρηθεί ότι απεικονίζει μία συνδρομική (concurrent) υπολογιστική δραστηριότητα, με την έννοια ότι όλες οι εκφράσεις που βρίσκονται στο δεξί μέλος μιας τέτοιας εντολής υπολογίζονται ταυτόχρονα. Μία τέτοια εντολή, η οποία αναπαριστά επακριβώς την έννοια της συστολικής επεξεργασίας, έχει, συνήθως, την ακόλουθη μορφή:

$$x, y := f(x^o, y^o), g(x^o, y^o),$$

και υποδηλώνει την εκχώρηση των συναρτήσεων $f(x^o, y^o)$, $g(x^o, y^o)$ στις μεταβλητές x και y, αντίστοιχα, όπου τα x^o, y^o είναι οι τιμές των x και y πριν από την εκτέλεση

της εντολής. Στο δεξί μέλος αυτών των εντολών εκχώρησης είναι δυνατόν να εμφανίζονται και εντολές υπό συνθήκη. Έτσι, μία εντολή υπό συνθήκη της μορφής:

$$x := \begin{cases} 0, & \text{if } a \geq 0 \\ 1, & \text{if } a < 0 \end{cases}$$

μπορεί να γραφεί ως εξής:

$x := 0$ ***if*** $a \geq 0$ ***or*** 1 ***if*** $a < 0$

Ένα πρόγραμμα που αναπαριστά ένα συστολικό αλγόριθμο αποτελείται από τις δηλώσεις των μεταβλητών του, την εκχώρηση αρχικών τιμών σε αυτές τις μεταβλητές και μία εντολή πολλαπλής εκχώρησης. Η εκτέλεση αυτής της εντολής επαναλαμβάνεται έως ότου το αριστερό και το δεξί μέλος της να αποκτήσουν την ίδια τιμή, γεγονός το οποίο υπονοεί ότι δεν είναι εφικτή καμία περαιτέρω αλλαγή στις τιμές των μεταβλητών.

Προκειμένου να διασπαστεί μία εντολή πολλαπλής εκχώρησης στις επιμέρους εντολές που την απαρτίζουν, έτσι ώστε να είναι εύκολη η κατανόησή της, χρησιμοποιείται το σύμβολο ||. Για παράδειγμα, η έκφραση $x, y := a, b$ είναι ισοδύναμη με την έκφραση $x := a \,||\, y := b$. Ο συμβολισμός

$< i \text{ in } S :: \,||\, Q(i) >$

όπου S είνα ένα σύνολο και $Q(i)$ είναι μία εκχώρηση (ή πολλαπλή εκχώρηση), υποδηλώνει μία εντολή η οποία προκύπτει αντικαθιστώντας το δείκτη i μέσα στο $Q(i)$ με κάθε στοιχείο του S. Για παράδειγμα, η έκφραση

$< i \text{ in } 0..1 :: \,||\, X[i] := Y[i] >$

είναι ισοδύναμη με τις εκφράσεις

$|| \; X[0] := Y[0]$
$|| \; X[1] := Y[1]$

Κεφάλαιο 6

Συμβολική Αναπαράσταση Συστολικών Αλγορίθμων

6.1 Εισαγωγή

Η εύρεση ικανοποιητικών μεθόδων αναπαράστασης των συστολικών αλγορίθμων έχει απασχολήσει ιδιαίτερα την επιστημονική κοινότητα την τελευταία 15-ετία και πολλές προσπάθειες έχουν γίνει προς την κατεύθυνση αυτή. Συχνά, οι μέθοδοι αυτές οδηγούν στην εξάλλειψη του πλεονάζοντος χρόνου υλοποίησης που απαιτούν, συνήθως, οι περισσότεροι ευρετικοί συστολικοί αλγόριθμοι, κάτι το οποίο αποτελεί και το βασικότερο στόχο ανάπτυξης των συγκεκριμένων μεθόδων. Όπως ήδη αναφέρθηκε, η χρήση του εργαλείου SDT επιτρέπει την εφαρμογή μιας συστηματικής μεθοδολογίας αναπαράστασης συστολικών αλγορίθμων, μέσω της οποίας διασφαλίζεται, τόσο η κανονικότητα της ροής των δεδομένων, όσο και η παρουσίαση των πιθανώς πολλαπλών κατευθύνσεων αυτής της ροής στον τρισδιάστατο χώρο, γεγονός το οποίο βοηθάει σημαντικά στην παρακολούθηση της υπολογιστικής δραστηριότητας του εκάστοτε αλγόριθμου. Επιπλέον, το εργαλείο αυτό θέτει τις προυποθέσεις για την ανάπτυξη ορθών συστολικών αλγορίθμων, απαλλαγμένων από λάθη που πηγάζουν από την αυθαίρετη παρουσίασή τους κάνοντας χρήση διαγραμμάτων, τα οποία απεικονίζουν ένα σύνολο από διατεταγμένα συστολικά κελιά, τις μεταξύ τους διασυνδέσεις, τη ροή των

δεδομένων και την επεξεργασία που λαμβάνει χώρα. Η πρακτική χρησιμότητα του συγκεκριμένου εργαλείου θα περιγραφεί με την εφαρμογή του για τη συμβολική αναπαράσταση δύο αποδοτικών συστολικών αλγορίθμων επίλυσης τριδιαγώνιων γραμμικών συστημάτων, οι οποίοι είναι, ο αλγόριθμος *Αναδιεύθυνσης της Κεντρικής Διαγωνίου* (Main Diagonal Redirection - MDR, βλέπε Bekakos, et al, [17]) και ο αλγόριθμος της *προς τα Πίσω και Εμπρός Απαλοιφής και Επίλυσης* (Backward & Forward - B&F, βλέπε Bekakos, et al, [20]).

Η επίλυση τριδιαγώνιων γραμμικών συστημάτων είναι ένα πολύ σημαντικό έργο, γιατί τέτοιου είδους συστήματα απαντώνται συχνά κατά την επίλυση κανονικών και μερικών διαφορικών εξισώσεων, ενώ αποτελούν, επίσης, το τελικό προιόν πολλών ορθογωνίων τεχνικών ελαχιστοποίησης (orthogonal reduction techniques), οι οποίες χρησιμοποιούνται για την αποτίμηση των ιδιοτιμών και των ιδιοδυανυσμάτων πινάκων.

6.2 Χωροαποδοτικός Αλγόριθμος Αναδιεύθυνσης της Κεντρικής Διαγωνίου (Area Efficient MDR - AEMDR)

Ας θεωρήσουμε, χωρίς απώλεια της γενικότητας, το γραμμικό, συμμετρικό και τριδιαγώνιο σύστημα της μορφής

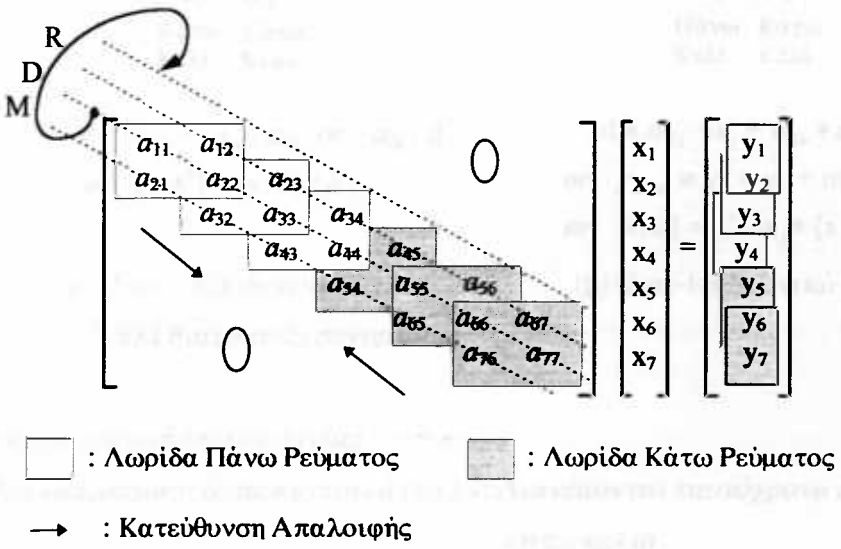
$$A\,x = y, \tag{6.2:1}$$

όπου τα x και y είναι τα διανύσματα των αγνώστων και των σταθερών όρων, αντίστοιχα, και A είναι ένας (nxn) τριδιαγώνιος και συμμετρικός πίνακας με την ακόλουθη μορφή

$$A = \begin{bmatrix} a_{11} & a_{12} & & & 0 \\ a_{21} & a_{22} & a_{23} & & \\ & \ddots & \ddots & \ddots & \\ 0 & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

Εφαρμόζοντας την προσέγγιση AEMDR, τα στοιχεία της κεντρικής διαγωνίου επανατοποθετούνται σε μία νέα διαγώνιο πάνω από την υπερδιαγώνιο

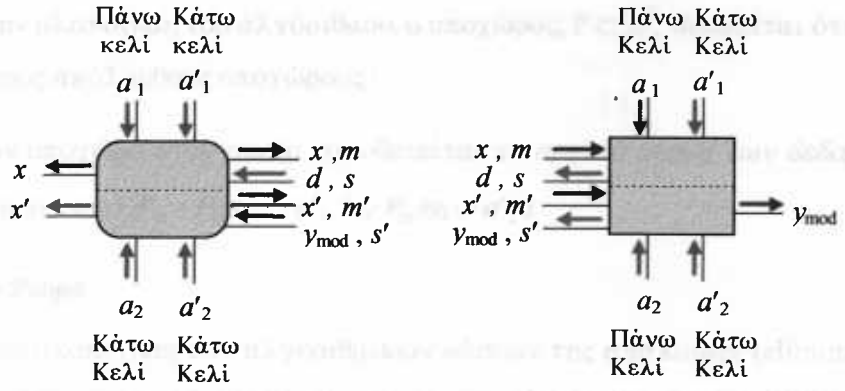
(superdiagonal) του αρχικού πίνακα. Έτσι, το γραμμικό σύστημα προς επίλυση αποκτά τη μορφή του σχήματος 6.2-1, στο οποίο περιγράφεται και το αντίστοιχο ρεύμα των δεδομένων που εισερχεται στη συστολική διάταξη. Η συγκεκριμένη μέθοδος βελτιώνει, τόσο τη συνολική χρονική πολυπλοκότητα, η οποία ισούται με 2n+1 χρονικά βήματα, όσο και το βαθμό χρήσης των συστολικών κελιών. Ένα επιπλέον πλεονέκτημα της μεθόδου είναι ότι χρησιμοποιεί (w-1) επεξεργαστές, όπου w=p+q-1 και p, q τα ημι-εύρη ζώνης του πίνακα A.



Σχήμα 6.2-1: Ρεύματα Δεδομένων σε έναν Ταινιωτό Πίνακα διαστάσεων (7x7)

Η συστολική διάταξη που χρησιμοποιεί ο αλγόριθμος AEMDR αποτελείται από ένα ‹2-σε-1› εξειδικευμένο κελί το οποίο πραγματοποιεί την πράξη της διαίρεσης (*συνοριακό* κελί), και από ένα ‹2-σε-1› κανονικό κελί. Η αρχιτεκτονική καθενός από αυτά τα κελιά επιτρέπει την ταυτόχρονη εκτέλεση δύο IPS πράξεων, εξασφαλίζοντας μία κανονική ροή του ρεύματος των δεδομένων σε διαδοχικά χρονικά βήματα. Μία αφηρημένη περιγραφή, τόσο της μορφής, όσο και της λειτουργίας αυτών των κελιών στο διδιάστατο χώρο, δίνεται στο σχήμα 6.2-2. Στα διακεκομένα τετράγωνα προσδιορίζονται οι πράξεις για τη φάση της *προς τα πίσω αντικατάστασης*. Η συνολική χρονική πολυπλοκότητα για τη λύση του τριδιαγώνιου γραμμικού συστήματος ισούται με 2n+1 χρονικά βήματα, όταν τα x_i's εξέρχονται από τις γραμμές επικοινωνίας των εξειδικευμένων κελιών διαίρεσης.

Από τη λειτουργία των κελιών καθίσταται προφανές ότι κατά τη διάρκεια των πρώτων *n* χρονικών βημάτων το συνοριακό κελί υπολογίζει τον πολλαπλασιαστή



$m_{ik} = - a_{ik} / a_{kk}$ or $- a_{ik} / d$

or $\{x, x'\} = \{s, s'\} / a'_{ij}$

(α) ‹2-σε-1› εξειδικευμένο κελί διαίρεσης (*συνοριακό*)

$d \equiv a'_{ij} = a_{ij} + m_{ik} * a_{kj}$

or $y_{mod} \equiv y_i' = y_i + m_{ik} * y_k$

or $\{s, s'\} = y_i' - a_{ij} * \{x, x'\}$

(β) ‹2-σε-1› κανονικό IPS κελί

ενεργή γραμμή επικοινωνίας : ⟶

Πολλαπλασιαστής m, κεντρικά (x,y)'s: Διανέμονται ταυτόχρονα στα πάνω και κάτω κελιά

Σχήμα 6.2-2: Περιγραφή Λειτουργίας Συστολικών Κελιών

m_{ik}, δηλαδή, εκτελεί την πράξη $m_{ik} = - a_{ij} / a'_{kj}$, όπου i>k ή i< k και k = j. Στη συνέχεια, το κελί υπολογίζει το διάνυσμα λύσης, δηλαδή., εκτελεί την πράξη $\{x , x'\} = \{s , s'\} / a'_{ij}$, όπου i = j. Από την άλλη μεριά, κατά τη διάρκεια της ίδιας χρονικής περιόδου, στο πάνω τμήμα του κανονικού IPS κελιού τροποποιείται ένα στοιχείο της κεντρικής διαγωνίου του πίνακα A σε κάθε διακεκριμένο χρονικό βήμα, δηλαδή., εκτελείται η πράξη $a'_{ij} = a_{ij} + m_{ik} * a_{kj}$, όπου i = j και k< j ή k>j, ενώ στο κάτω τμήμα του ίδιου κελιού εκτελείται, ταυτόχρονα, η πράξη $y_i = y_i + m_{ik} * y_k$. Στη συνέχεια, το κελί υπολογίζει τα μερικά γινόμενα s_i, όπου $\{s , s'\} = y_i' - a_{ij} * \{x , x'\}$ και i=1,...,n.

6.2.1 Μονόδρομη Αναπαράσταση στο Χώρο

Για την υλοποίηση του αλγόριθμου, ο υποχώρος, $P \subset Z^3$, θεωρείται ότι αποτελείται από τους ακόλουθους υποχώρους:

- Τον υποχώρο στον οποίο τοποθετείται το *αρχικό ρεύμα των δεδομένων* (initial data stream) $P_{in} = P_{in}(a_1 \cup a'_1) \cup P_{in}(a_2 \cup a'_2)$.

Πάνω Ρεύμα

Για την υλοποίηση των αλγοριθμικών φάσεων της *απαλοιφής* (elimination) και της *τροποποίησης του διανύσματος των σταθερών όρων* (r.h.s. modification), το πάνω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a_1 \cup a'_1) = \{(i, j, 0) \mid 1 < i \leq r, 1 \leq j \leq 2\}$, δηλαδή., στο επίπεδο k=0 (i,j>0), και $P_{in}(a_2 \cup a'_2) = \{(0, j, k) \mid 1 \leq j \leq r, 1 \leq k < r\}$, δηλαδή., στο επίπεδο i=0 (j,k>0), όπου πραγματοποιούνται, αντίστοιχα, οι εκχωρήσεις $a_1(i, j, 0) := a_{ij}$, $a_2(0, j, k) := a_{kj}$, $a'_1(i, 2, 0) := y_i$ and $a'_2(0, j, k) := y_k$ χρησιμοποιώντας μία από τις επιτρεπτές $d\Diamond p_s$. Οι εκχωρήσεις αυτές προσδιορίζονται από τη διαμόρφωση του αρχικού πάνω ρεύματος των δεδομένων σύμφωνα με τις ακόλουθες σχέσεις:

$a_1(i, 1, 0) := a_{ij}$, $i > j, 1 \leq j < r$

$a_1(i, 2, 0) := a_{ij}$, $i = j, 1 < j \leq r$

$$a_2(0, j, k) := \begin{cases} a_{kj}, & k = j = 1 \quad \{\textit{συνοριακό κελί}\} \\ a_{kj}, & k < j, 1 < j \leq r \quad \{\textit{κανονικό κελί}\} \end{cases}$$

$a'_1(i, 2, 0) := y_i$, $1 < i \leq r$

$$a'_2(0, k+1, k) := \begin{cases} y_k, & k = 1 \quad \{\textit{κανονικό κελί}\} \\ p_k, & 1 < k < r \quad \{\textit{κανονικό κελί}\}, \end{cases}$$

όπου $r = \left\lceil \frac{n}{2} \right\rceil$.

Η παράμετρος p_k ορίζει ένα σύνολο μεταβλητών, για $1 < k < n$, οι οποίες χρησιμοποιούνται για την εκ των προτέρων δέσμευση συγκεκριμένων θέσεων στο ρεύμα των δεδομένων. Αυτό συνεπάγεται ότι τα τροποποιημένα στοιχεία του

διανύσματος y, τα οποία εξέρχονται από το αντίστοιχο κελί, συλλέγονται καθ' οδόν *(on-the-fly)* κάθε φορά και επανατοποθετούνται στο ρεύμα των δεδομένων με σκοπό να εισαχθούν εκ νέου στο ίδιο κελί, προκειμένου να χρησιμοποιηθούν για την τροποποίηση άλλων στοιχείων του ίδιου διανύσματος. Ο δείκτης k αντιστοιχεί στο δείκτη του τροποποιημένου στοιχείου, το οποίο επανατοποθετείται στο ρεύμα των δεδομένων.

Για την υλοποίηση της αλγοριθμικής φάσης της *προς τα πίσω αντικατάστασης* (back substitution), το πάνω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a_1) = \{(i^*, j, 0) \mid r < i^* \le n+1, 1 \le j \le 2\}$, και $P_{in}(a_2) = \{(0, j^*, k^*) \mid r < j^* \le n, r \le k^* < n\}$, όπου $i^* = n-i+1$, $j^* = n-j+1$ και $k^* = n-k+1$, ενώ οι αντίστοιχες εκχωρήσεις που προσδιορίζονται από το αρχικό πάνω ρεύμα των δεδομένων είναι:

$$a_1(i^*+1, 1, 0) := \begin{cases} a_{ij}, & i = j = 1 \\ p_{ij}, & i = j, 1 < i < r \end{cases}$$

$$a_1(i^*, 2, 0) := a_{ij}, \; i < j, 1 < j \le r$$

$$a_2(0, i^*, i^*-1) := \begin{cases} y_i, & i = 1 \quad \{\textit{κανονικό κελί}\} \\ p_i, & 1 < i < r \quad \{\textit{κανονικό κελί}\} \end{cases}$$

Η παράμετρος p_{ij} προσδιορίζει επίσης ένα σύνολο μεταβλητών, για $1 < i, j < n$ και $i = j$, και χρησιμοποιείται κατά τον ίδιο τρόπο όπως και η παράμετρος p_k, με τη διαφορά ότι οι μεταβλητές αυτές αντιστοιχούν στα τροποποιημένα στοιχεία της κεντρικής διαγωνίου του πίνακα Α.

Κάτω Ρεύμα

Για την υλοποίηση των αλγοριθμικών φάσεων της *απαλοιφής* και της *τροποποίησης του διανύσματος των σταθερών όρων*, το κάτω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a_1 \cup a'_1) = \{(i^*, j, -1) \mid 1 < i^* \le r, 1 \le j \le 2\}$, δηλαδή., στο επίπεδο $k = -1$ $(i, j > 0)$, και $P_{in}(a_2 \cup a'_2) = \{(-1, j^*, k^*) \mid 1 \le j^* \le r, \quad 1 \le k^* < r\}$, δηλαδή., στο επίπεδο $i = -1 (j, k > 0)$, όπου πραγματοποιούνται, αντίστοιχα, οι εκχωρήσεις $a_1(i^*, j, -1) := a_{ij}$, $a_2(-1, j^*, k^*) := a_{kj}$, $a'_1(i^*, 2, -1) := y_i$ και $a'_2(-1, k^*+1, k^*) := y_k$.

Οι εκχωρήσεις αυτές προσδιορίζονται από τη διαμόρφωση του αρχικού κάτω ρεύματος των δεδομένων σύμφωνα με τις ακόλουθες σχέσεις:

$a_1(i^*, 1, -1) := a_{ij},\ i < j,\ r < j \le n$

$a_1(i^*, 2, -1) := a_{ij},\ i = j,\ r < j < n$

$$a_2(-1, j^*, k^*) := \begin{cases} a_{kj},\ k = j = n & \{\ \textit{συνοριακό κελί}\ \} \\ a_{kj},\ k > j,\ r \le j < n & \{\ \textit{κανονικό κελί}\ \} \end{cases}$$

$a'_1(i^*, 2, -1) := y_i,\ r < i < n$

$$a'_2(-1, k^*+1, k^*) := \begin{cases} y_k,\ k = n & \{\ \textit{κανονικό κελί}\ \} \\ p_k,\ r < k < n & \{\ \textit{κανονικό κελί}\ \} \end{cases}$$

Για την υλοποίηση της αλγοριθμικής φάσης της *προς τα πίσω αντικατάστασης*, το κάτω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a'_1) = \{(i, j, 0) \mid r < i \le n+1,\ 1 \le j \le 2\}$, και $P_{in}(a'_2) = \{(0, j, k) \mid r < j \le n,\ r \le k < n\}$, ενώ οι αντίστοιχες εκχωρήσεις που προσδιορίζονται από το αρχικό κάτω ρεύμα των δεδομένων είναι:

$$a'_1(i+1, 1, 0) := \begin{cases} a_{ij},\ i = j = n \\ p_{ij},\ i = j,\ r < i < n \end{cases}$$

$a'_1(i, 2, 0) := a_{ij},\ i > j,\ r \le j < n$

$$a'_2(0, i, i-1) := \begin{cases} y_i,\ i = n & \{\ \textit{κανονικό κελί}\ \} \\ p_i,\ r < i < n & \{\ \textit{κανονικό κελί}\ \} \end{cases}$$

Στη συγκεκριμένη φάση του αλγόριθμου, το πάνω και κάτω ρεύμα δεδομένων υφίστανται επεξεργασία ταυτόχρονα από το πάνω και κάτω τμήμα, αντίστοιχα, του ίδιου κάθε φορά κελιού. Κατά συνέπεια, τα στοιχεία του κάτω ρεύματος των δεδομένων τοποθετούνται στα επίπεδα k=0 (i,j>0) και i=0 (j,k>0), αντί των επιπέδων k = -1 (i,j>0) και i = -1 (j,k>0).

- Τον υποχώρο των *εσωτερικών υπολογισμών* (inner computations) $P_{int} = \{(i, j, k) \mid 1 < i \le n+1,\ 1 \le j \le 2,\ k=1\}$. Κατά τη διάρκεια των πρώτων n χρονικών βημάτων, οι

υπολογισμοί που σχετίζονται με κάθε σημείο αυτού του υποχώρου, για το οποίο ισχύει ότι j=1 (συνοριακό κελί), είναι:

$m_{ik} := -a_{ij} / a_{kj}$ ή $m_{ik} := -a_{ij} / d\{\equiv a'_{kj}\}$, $i > k$ ή $i < k$, $k = j$

δηλαδή,

$m(i, j, k) := -a_1(i, j, k) / a_2(i, j, k)$ **if** $t \leq 2$ **or**

$-a_1(i, j, k) / d(i, j, k)$ **if** $2 < t < n$

Κατά τη διάρκεια των τελευταίων n χρονικών βημάτων, οι αντίστοιχοι υπολογισμοί, οι οποίοι πραγματοποιούνται στο ίδιο PE, είναι:

$x_i := s_i / a'_{ij}$ { *πάνω κελί* } και $x'_i := s'_i / a'_{ij}$ { *κάτω κελί* }

δηλαδή,

$x(i, j, k) := s(i, j, k) / a_1(i, j, k)$ **if** $t > n+1$ { *πάνω κελί* }

$x'(i, j, k) := s'(i, j, k) / a'_1(i, j, k)$ **if** $t > n+1$ { *κάτω κελί* }

Παρόμοια, κατά τη διάρκεια των πρώτων n χρονικών βημάτων, οι υπολογισμοί που σχετίζονται με κάθε σημείο αυτού του υποχώρου, για j = 2 (κανονικό κελί), είναι:

$d \equiv a'_{ij} := a_{ij} + m_{ik} * a_{kj}$, $i = j$, $k < j$ ή $k > j$ { *πάνω κελί* }

$y_{mod} \equiv y'_i := y_i + m_{ik} * y_k$, $i > k$ ή $i < k$ { *κάτω κελί* }

δηλαδή,

$d(i, j, k) := a_1(i, j, k) + m(i, j, k) * a_2(i, j, k)$ **if** $t \leq n$ και

$y_{mod}(i, j, k) := a'_1(i, j, k) + m'(i, j, k) * a'_2(i, j, k)$ **if** $t \leq n$

ενώ, κατά τη διάρκεια των τελευταίων n χρονικών βημάτων, οι υπολογισμοί που σχετίζονται με το ίδιο PE είναι:

$s := y'_i - a_{ij} * x$, $i < j$ { *πάνω κελί* } και

$s' := y'_i - a_{ij} * x'$, $i > j$ { *κάτω κελί* }

δηλαδή,

$s(i, j, k) := a_2(i, j, k) - x(i, j, k) * a_1(i, j, k)$ **if** $t > n+1$

$s'(i, j, k) := a'_2(i, j, k) - x'(i, j, k) * a'_1(i, j, k)$ **if** $t > n+1$

• Τον υποχώρο των *αποτελεσμάτων εξόδου* (output results) $P_{out} = \{(n+2, j, k) \mid r \leq j, k \leq n\}$, όπου πραγματοποιούνται οι ακόλουθες εκχωρήσεις, οι οποίες αντιστοιχούν στο διάνυσμα λύσης του συστήματος, $\vec{x}$:

$$x_i = \left\{ \begin{array}{l} x(n+2, j, k) := x(n+2-m, j, k), \quad i = m \quad \{\textit{πάνω συνοριακό κελί}\} \\ x'(n+2, j, k) := x'(n+2-m, j, k), \quad i = n-m+1 \quad \{\textit{κάτω συνοριακό κελί}\} \end{array} \right\}$$

για $i=1,\ldots,n$, $m = r, r-1, \ldots, 1$, και $t = n + 2m$, όταν το n είναι *περιττό*.

Η αναπαράσταση των υπολογισμών στο χώρο είναι εφικτή μόνον για τις $d\Diamond p_s$ (1,1,0), (1,0,1), (0,1,1) και (1,1,1). Οι $d\Diamond p_s$ (1,0,0), (0,1,0) και (0,0,1) δεν αποτελούν επιτρεπτές κατευθύνσεις προβολής, γιατί σε οποιαδήποτε από αυτές ένα τμήμα του χώρου P_{int} επικαλύπτεται από ένα τμήμα του χώρου P_{in}.

Κατά συνέπεια, θα πρέπει να σημειωθεί ότι ο χώρος P_{in} $(a_2 \cup a'_2)$ είναι στο επίπεδο i=0 (j,k >0) για τις $d\Diamond p_s$ (1,1,0), (1,0,1) και (1,1,1). Ο χώρος αυτός στον οποίο τοποθετούνται τα αρχικά δεδομένα είναι η γραμμή k=1, του επιπέδου i=0, για την $d\Diamond p$ (1,1,0), οι γραμμές j=1 και j=2, του επιπέδου i=0, για την $d\Diamond p$ (1,0,1), και το επίπεδο i=0 (j,k >0), για την $d\Diamond p$ (1,1,1). Όσον αφορά στην $d\Diamond p$ (0,1,1), τα στοιχεία της μήτρας a_{kj}, για k<j ή k>j, y_k ή y_i, εισέρχονται στη συστολική διάταξη από το επίπεδο j = 0 (i,k >0), αντί από το επίπεδο i = 0 (j,k>0), όπως συμβαίνει στις τρεις προηγούμενες $d\Diamond p_s$.

Επιπλέον, η εφαρμογή της τεχνικής της Περιστροφής και Αναδίπλωσης (Rotate & Fold - R&F, βλέπε Bekakos, [16]) επιβάλλει την επέκταση του πρώτου οκτάεδρου του χώρου, στο οποίο θεωρούμε ότι υλοποιείται ο αλγόριθμος, έτσι ώστε να συμπεριληφθούν και τα επίπεδα i = -1 (j,k >0), j = -1 (i,k>0) και k = -1 (i,j>0), στα οποία τοποθετείται, όπως ήδη αναφέρθηκε, το αρχικό κάτω ρεύμα δεδομένων για την υλοποίηση της φάσης της απαλοιφής και της φάσης τροποποίησης του διανύσματος των σταθερών όρων.

6.2.2 Προγραμματιστική Αναπαράσταση στο Χώρο

Στην Παράγραφο αυτή παρουσιάζεται η προγραμματιστική αναπαράσταση του επαναληπτικού συστολικού αλγόριθμου AEMDR, κάνοντας χρήση μιας κατάλληλης συμβολικής προσέγγισης και της συγκεκριμένης κατεύθυνσης

προβολής (1,1,0), ενώ θα πρέπει να σημειωθεί ότι ένα παρόμοιο πρόγραμμα μπορεί να δοθεί για όλες τις επιτρεπτές κατευθύνσεις προβολής. Επίσης, θα πρέπει να ληφθεί υπόψη ότι όλες οι εκχωρήσεις οι οποίες πραγματοποιούνται και από τα δύο υπολογιστικά ρεύματα, εκτελούνται παράλληλα, γεγονός το οποίο υπονοείται έμμεσα σε αυτή τη φάση της προγραμματιστικής αναπαράστασης.

Algorithm_AEMDR

t:=1

for *i:=2* ***to*** *n+1* ***do***

for *j:=i-1* ***to*** *i* ***do***

if *j<i* ***then*** *{συνοριακό κελί}*

$a_1(i,j,k) := a_1(i,j,0)$

$a_2(i,j,k) := a_2(0,j,k)$ ***if*** $t \leq 2$

$d(i,j,k) := d(i-1,j,k)$ ***if*** $((1 < t < n) \vee (t = n+1))$

$m(i,j,k) := -a_1(i,j,k) / a_2(i,j,k)$ ***if*** $t \leq 2$ ***or***

$-a_1(i,j,k) / d(i,j,k)$ ***if*** $2 < t < n$

$s(i,j,k) := s(i-1,j,k)$ ***if*** $t > n+1$

$s'(i,j,k) := y_{mod}(i-1,j,k)$ ***if*** $t = n+1$ ***or***

$s'(i-1,j,k)$ ***if*** $t > n+1$

$x(i,j,k) := s'(i,j,k) / d(i,j,k)$ ***if*** $t = n+1$ ***or***

$s(i,j,k) / a_1(i,j,k)$ ***if*** $t > n+1$

$a'_1(i,j,k) := a'_1(i,j,0)$ ***if*** $t > n+1$

$x'(i,j,k) := s'(i,j,k) / a'_1(i,j,k)$ ***if*** $t > n+1$

else if *j=i* ***then*** *{normal cell}*

$a_1(i,j,k) := a_1(i,j,0)$ ***if*** $((t < n) \vee (t > n+1))$ ***or***

$d(i,j,k)$ ***if*** $t=n$

$a_2(i,j,k) := a_2(0,j,k)$

$m(i,j,k) := m(i,j-1,k)$ ***if*** $t \leq n$

$d(i,j,k) := a_1(i,j,k) + m(i,j,k) * a_2(i,j,k)$ ***if*** $t \leq n$

$x(i,j,k) := x(i,j-1,k)$ ***if*** $t > n+1$

$s(i,j,k) := a_2(i,j,k) - x(i,j,k) * a_1(i,j,k)$ ***if*** $t > n+1$

$a'_1(i,j,k) := a'_1(i,j,0)$ **if** $((t < n) \wedge (t > n+1))$ **or**

$y_{mod}(i,j,k)$ **if** $t=n$

$a'_2(i,j,k) := a'_2(0,j,k)$

$m'(i,j,k) := m(i,j-1,k)$ **if** $t \leq n$

$y_{mod}(i,j,k) := a'_1(i,j,k) + m'(i,j,k) * a'_2(i,j,k)$ **if** $t \leq n$

$x'(i,j,k) := x'(i,j-1,k)$ **if** $t > n+1$

$s'(i,j,k) := a'_2(i,j,k) - x'(i,j,k) * a'_1(i,j,k)$ **if** $t > n+1$

$t := t+1$

if $t=n$ **then** { *κάποια εικονική ή ψευδο-λειτουργία* (dummy operation) *μόνο για το πάνω ρεύμα* }

Μέσω της παραπάνω προγραμματιστικής αναπαράστασης, οι υπολογισμοί φαίνεται να εκτείνονται στον τρισδιάστατο χώρο, ενώ όλα τα δεδομένα εισόδου, τα οποία εκχωρούνται στις μεταβλητές εισόδου a_1, a_2, a'_1 και a'_2, έχουν τοποθετηθεί στο επεκτεταμένο, αρχικά θεωρούμενο, οκτάεδρο του χώρου Z^3.

Εξαιτίας της εφαρμογής της τεχνικής R&F, ο συγκεκριμένος αλγόριθμος εφαρμόζεται ταυτόχρονα και στο πάνω, αλλά και στο κάτω ρεύμα δεδομένων, τα οποία διεμπλέκονται καθώς υφίστανται επεξεργασία κατά τη διάρκεια της φάσης της απαλοιφής και της φάσης της τροποποίησης του διανύσματος των σταθερών όρων, με το πάνω ρεύμα να προηγείται κατά ένα χρονικό βήμα. Το γεγονός αυτό συνεπάγεται ότι, όταν τα δεδομένα εισόδου που αντιστοιχούν στο πάνω ρεύμα εισέρχονται στη συστολική διάταξη, τα αντίστοιχα δεδομένα εισόδου του κάτω ρεύματος μετακινούνται από τα επίπεδα k = -1 (i,j>0) και i = -1 (j,k>0), στα επίπεδα k=0 (i,j>0) και i=0 (j,k>0), προκειμένου να εισέλθουν στα PEs κατά το επόμενο χρονικό βήμα.

6.2.2.1 Εξαρτήσεις στο Γράφημα Υπολογιστικής Δραστηριότητας

Η διάταξη των υπολογισμών στο χώρο P, για τον αλγόριθμο AEMDR, σε σχέση με την προγραμματιστική του αναπαράσταση, είναι δυνατό να περιγραφεί από ένα γράφημα καθολικών εξαρτήσεων της μορφής $\Gamma_g = (P_{int}, \vec{e}^3_{\alpha_1}, \vec{Q}^3_2, \vec{e}^3_m, \vec{e}^3_{\alpha'_1}, \vec{Q}'^3_2, \vec{e}^3_{m'})$,

όπου τα διανύσματα $\vec{Q}_2^3(i,0,0)$, $\vec{Q}_2'^3(i,0,0)$ είναι τα *καθολικά* διανύσματα εξαρτήσεων, ενώ τα διανύσματα $\vec{e}_{\alpha_1}^3(0,0,1)$, $\vec{e}_{\alpha_i'}^3(0,0,1)$, $\vec{e}_m^3(0,1,0)$, $\vec{e}_{m'}^3(0,1,0)$ είναι τα *τοπικά* διανύσματα εξαρτήσεων. Τα διανύσματα $\vec{e}_{\alpha_1}^3$, $\vec{e}_{\alpha_i'}^3$, $\vec{Q}_2^3$, $\vec{Q}_2'^3$ εκφράζουν, ουσιαστικά, τη συσχέτιση που υπάρχει μεταξύ του υπολογισμού που πραγματοποιείται στο σημείο $p(u, v, w) \in P_{int}(\pi_\rho)$ και των αντίστοιχων δεδομένων που εκχωρούνται στις κατάλληλες μεταβλητές εισόδου.

Προκειμένου για την απαλοιφή των καθολικών εξαρτήσεων για τα δεδομένα εισόδου a_{kj}, y_k ή y_i, θεωρούμε ότι

$$\varepsilon_2(p) = a_2(p\text{-}\vec{Q}_2^3) = \begin{cases} a_{kj}, & 1 \le j,k \le n \\ y_i, & 1 \le i < r \end{cases}$$

$$\varepsilon'_2(p) = a'_2(p\text{-}\vec{Q}_2'^3) = \begin{cases} y_k, & 1 \le j,k \le n \\ y_i, & r < i \le n, \end{cases}$$

όπου το $p \in P_{int}(\pi_\rho)$. Παρόμοια, το διάνυσμα λύσης, $\vec{x}$, λαμβάνεται θεωρώντας ότι

$\varepsilon_3(p) = x(p\text{-}\vec{Q}_2^3) = x_i, 1 \le i \le r$ { *πάνω ρεύμα* } και

$\varepsilon'_3(p) = x'(p\text{-}\vec{Q}_2'^3) = x_i, r < i \le n$ { *κάτω ρεύμα* }, όπου το $p \in P_{out}(\pi_\rho)$.

Εξαλείφοντας τις καθολικές εξαρτήσεις $\vec{Q}_2^3$, $\vec{Q}_2'^3$ το γράφημα Γ_g αντικαθίσταται από ένα ισοδύναμό του γράφημα $\Gamma_\ell = (P_{int}, \vec{e}_{\alpha_1}^3, \vec{e}_{\alpha_2}^3, \vec{e}_m^3, \vec{e}_{\alpha_i'}^3, \vec{e}_{\alpha_2'}^3, \vec{e}_{m'}^3)$, όπου τα διανύσματα $\vec{e}_{\alpha_2}^3(1,0,0)$, $\vec{e}_{\alpha_2'}^3(1,0,0)$ είναι τα τοπικά διανύσματα εξάρτησης για τα δεδομένα εισόδου a_{kj}, y_k ή y_i, αντίστοιχα. Το γράφημα που προέκυψε μπορεί, πλέον, να απεικονιστεί σε μια μονοδιάστατη συστολική διάταξη κάνοντας χρήση μιας επιτρεπτής $d\Diamond p$.

Απεικονίζοντας το γράφημα Γ_ℓ κατά μήκος της $d\Diamond p$ $\pi_\rho(\pi_i,\pi_j,\pi_k)$, προκύπτει ένας μονοδιάστατος συστολικός πίνακας (1D-SA) μεγέθους w-1. Ωστόσο, οι σωστές θέσεις των δεδομένων εισόδου στο προβολικό επίπεδο παρέχονται μόνον μέσω της επέκτασης του χώρου του αρχικού ρεύματος των δεδομένων, $P_{in}(\pi_\rho)$. Οι νέες θέσεις

των αρχικών δεδομένων εισόδου στο χώρο Z^3, λαμβάνονται με βάση την ακόλουθη σχέση, η οποία χρησιμοποιείται για κάθε ξεχωριστή dΦp.

$$\breve{p}_{\varepsilon} = position(p_{\varepsilon}) = \vec{p}_{\varepsilon} - [t(\breve{p}_{\cdot\varepsilon}) + f_g]\,\vec{e}^{\,3}_{\varepsilon}, \qquad (6.2.2.1{:}1)$$

$$f_g = \{ f_{t(op)}, f_{b(ottom)} \},\ \varepsilon = \{ a_1, a_2, a'_1, a'_2 \}$$

Η συνάρτηση χρόνου ορίζεται ως $t(p_{\varepsilon}) = i+j+k-4$, εφόσον ισχύει ότι $t(p_{start}) = p(2,1,1)$. Επιπλέον, θα πρέπει να σημειωθεί ότι, η διαδικασία επέκτασης εφαρμόζεται μόνον στο χώρο του αρχικού ρεύματος των δεδομένων και όχι στο χώρο των εσωτερικών υπολογισμών, $P_{int}(\pi_\rho)$.

6.2.3 Προβολικά Επίπεδα

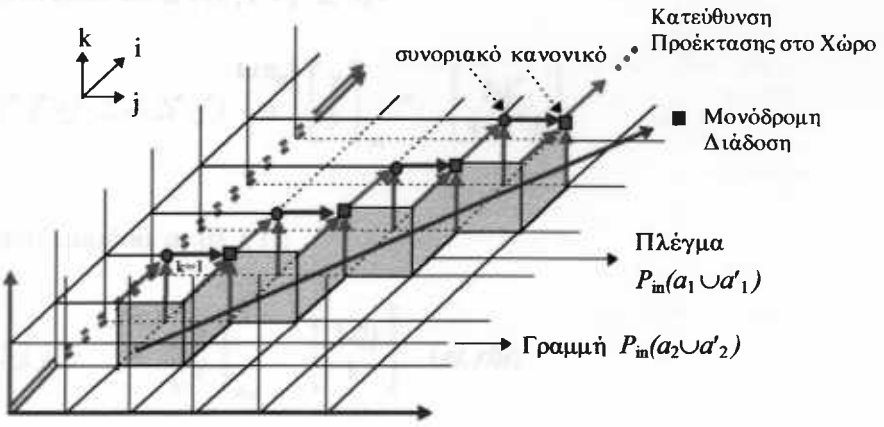
6.2.3.1 Κατεύθυνση Προβολής $\pi_\rho(1,1,0)$

Για την dΦp $\pi_\rho(1,1,0)$, ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση την απεικόνιση $P_{int} \ni p(i,j,k) \rightarrow p(u,v,w) \in P_{int}(\pi_\rho)$, όπου τα u, v, w προσδιορίζονται από τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$$u = u(i) = i, \qquad i = 2, \ldots, n+1$$
$$v = v(i,j) = i+j-2, \quad i = 2, \ldots, n+1,\ j = 1,2$$
$$w = w(k) = k, \qquad k = 1$$

Θα πρέπει να σημειωθεί ότι αυτή η απεικόνιση εφαρμόζεται, επίσης, και στα σημεία του χώρου $P_{in}(a_1 \cup a'_1)$, αλλά όχι σε αυτά του χώρου $P_{in}(a_2 \cup a'_2)$. Αυτή είναι μία σύμβαση, η οποία ισχύει για όλες τις επιτρεπτές $d\Phi p_s$ που θα εξεταστούν στη συνέχεια του παρόντος Κεφαλαίου. Για τη συγκεκριμένη dΦp, η συστολική διάταξη εκτείνεται στο επίπεδο k=1 (i,j>0), όπως φαίνεται στο σχήμα 6.2.3.1-1.

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την dΦp προκύπτουν από τη σχέση (6.2.2.1:1), όπου $f_{t_{el/rhs}} = 1$, $f_{t_{bs}} = 2$, $f_b = 2$ (el : φάση απαλοιφής / rhs : φάση τροποποίησης του διανύσματος των σταθερών όρων / bs : φάση προς τα πίσω αντικατάστασης).



Σχήμα 6.2.3.1-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την d◊p $\pi_\rho(1,1,0)$

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_1:

- d◊p $\pi_\rho(1,1,0)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

PE :

$$p(i, i+j-2, k) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2-j \\ k \end{bmatrix}$$

στοιχείο εισόδου a_1 (i, i+j-2, 0):

$$\breve{p}_{\alpha_1}(i, i+j-2, 5-2i-j) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1} = \begin{bmatrix} 2-j \\ 5-2i-j \end{bmatrix} \quad (el, rhs)$$

or

$$\breve{p}_{\alpha_1 \& \alpha'_1}(i^\#, i^\#+j^\#-2, 4-2i^\#-j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 2-j^\# \\ 4-2i^\#-j^\# \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_1 $(i^*, i^*+j^*-2, -1)$:

$$\check{p}_{\alpha'_1}\ (i^*, i^*+j^*-2, 4-2i^*-j^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_1} = \begin{bmatrix} 2-j^* \\ 4-2i^*-j^* \end{bmatrix}$$

στοιχείο εισόδου a_2 $(0, j, 1)$:

$$\check{p}_{\alpha_2}\ (2-j, j, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2} = \begin{bmatrix} 2-2j \\ 1 \end{bmatrix} \quad (el, rhs)$$

or

$$\check{p}_{\alpha_2 \,\&\, \alpha'_2}\ (1-j^{\#}, j^{\#}, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \,\&\, \alpha'_2} = \begin{bmatrix} 1-2j^{\#} \\ 1 \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_2 $(-1, j^*, 1)$:

$$\check{p}_{\alpha'_2}\ (1-j^*, j^*, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_2} = \begin{bmatrix} 1-2j^* \\ 1 \end{bmatrix}$$

όπου $i^{\#} = \{i, i^*\}$ και $j^{\#} = \{j, j^*\}$.

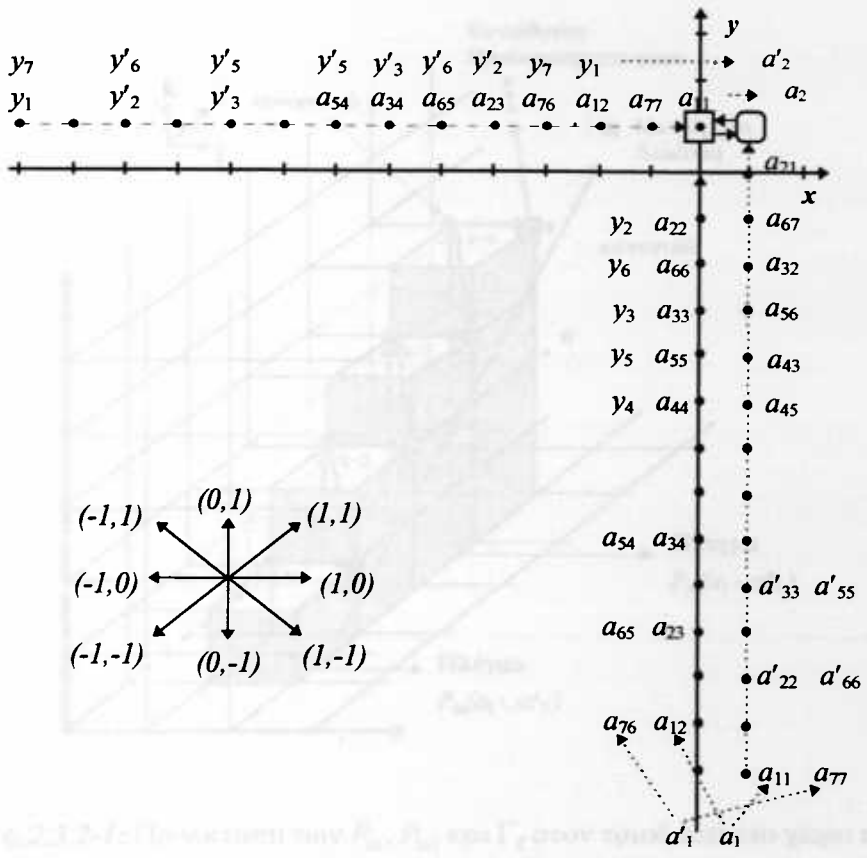
- ***Προβολικό επίπεδο***

Για την d◊p $\pi_\rho(1,1,0)$, η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.2.3.1-2.

6.2.3.2 Κατεύθυνση Προβολής $\pi_\rho(1,0,1)$

Για την d◊p $\pi_\rho(1,0,1)$, ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$$u = u(i) = i, \qquad i = 2, \ldots, n+1$$
$$v = v(j) = j, \qquad j = 1,2$$
$$w = w(i,k) = i+k-2, \quad i = 2, \ldots, n+1, \quad k = 1$$



Σχήμα 6.2.3.1-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

$(SA_1, d\Diamond p : \pi_\rho(1,1,0))$

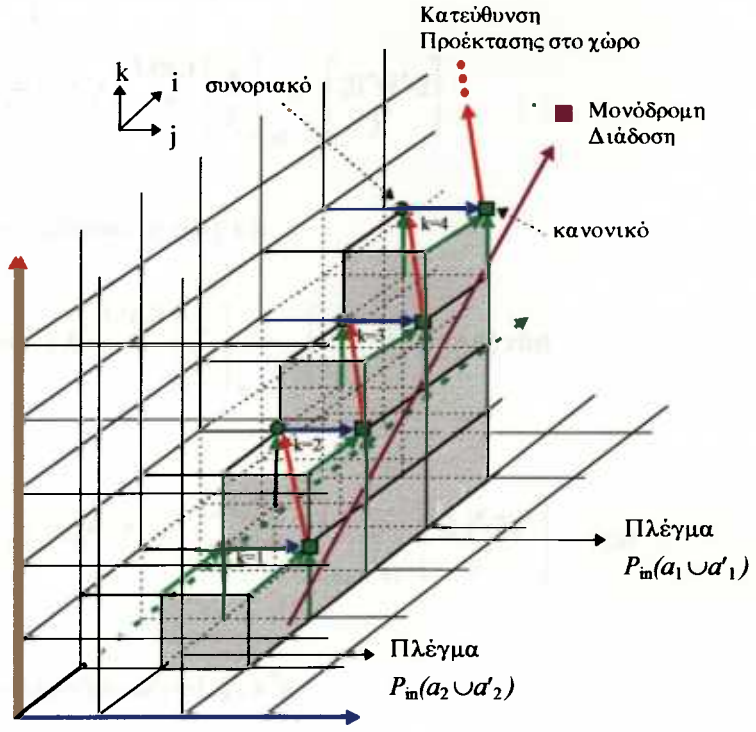
Γι' αυτή την d◊p η συστολική διάταξη εκτείνεται στα επίπεδα j=1 και j=2 (i,k>0), όπως φαίνεται στο σχήμα 6.2.3.2-1.

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την d◊p προκύπτουν από τη σχέση (6.2.2.1:1), όπου $f_{t_{el/rhs}} = 1$, $f_{t_{bs}} = 2$, $f_b = 2$.

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_2:

- d◊p $\pi_\rho(1,0,1)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$



Σχήμα 6.2.3.2-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την d◊p $\pi_\rho(1,0,1)$

PE :

$$p(i, j, i+k-2) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2\text{-}k \\ j \end{bmatrix}$$

στοιχείο εισόδου a_1 (i, j, 0):

$$\check{p}_{\alpha_1}(i, j, 3\text{-}i\text{-}j) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1} = \begin{bmatrix} 2i+j-3 \\ j \end{bmatrix} \quad (el, rhs)$$

or

$$\check{p}_{\alpha_1 \& \alpha'_1}(i^{\#}, j^{\#}, 2\text{-}i^{\#}\text{-}j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 2i^{\#}+j^{\#}-2 \\ j^{\#} \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_1 (i*, j*, -1):

$$\check{p}_{\alpha'_1}\ (i^*, j^*, 2\text{-}i^*\text{-}j^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_1} = \begin{bmatrix} 2i^*+j^*-2 \\ j^* \end{bmatrix}$$

στοιχείο εισόδου a_2 (0, j, k):

$$\check{p}_{\alpha_2}\ (3\text{-}j\text{-}k,\ j,\ k) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2} = \begin{bmatrix} 3\text{-}j\text{-}2k \\ j \end{bmatrix} \quad (el,\ rhs)$$

or

$$\check{p}_{\alpha_2 \,\&\, \alpha'_2}\ (2\text{-}j^{\#}\text{-}k^{\#},\ j^{\#},\ k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \,\&\, \alpha'_2} = \begin{bmatrix} 2\text{-}j^{\#}\text{-}2k^{\#} \\ j^{\#} \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_2 (-1, j*, k*):

$$\check{p}_{\alpha'_2}\ (2\text{-}j^*\text{-}k^*,\ j^*,\ k^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_2} = \begin{bmatrix} 2\text{-}j^*\text{-}2k^* \\ j^* \end{bmatrix}$$

όπου $k^{\#} = \{ k, k^* \}$.

- ***Προβολικό επίπεδο***

Για την dθρ $\pi_\rho(1,0,1)$, η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.2.3.2-2.

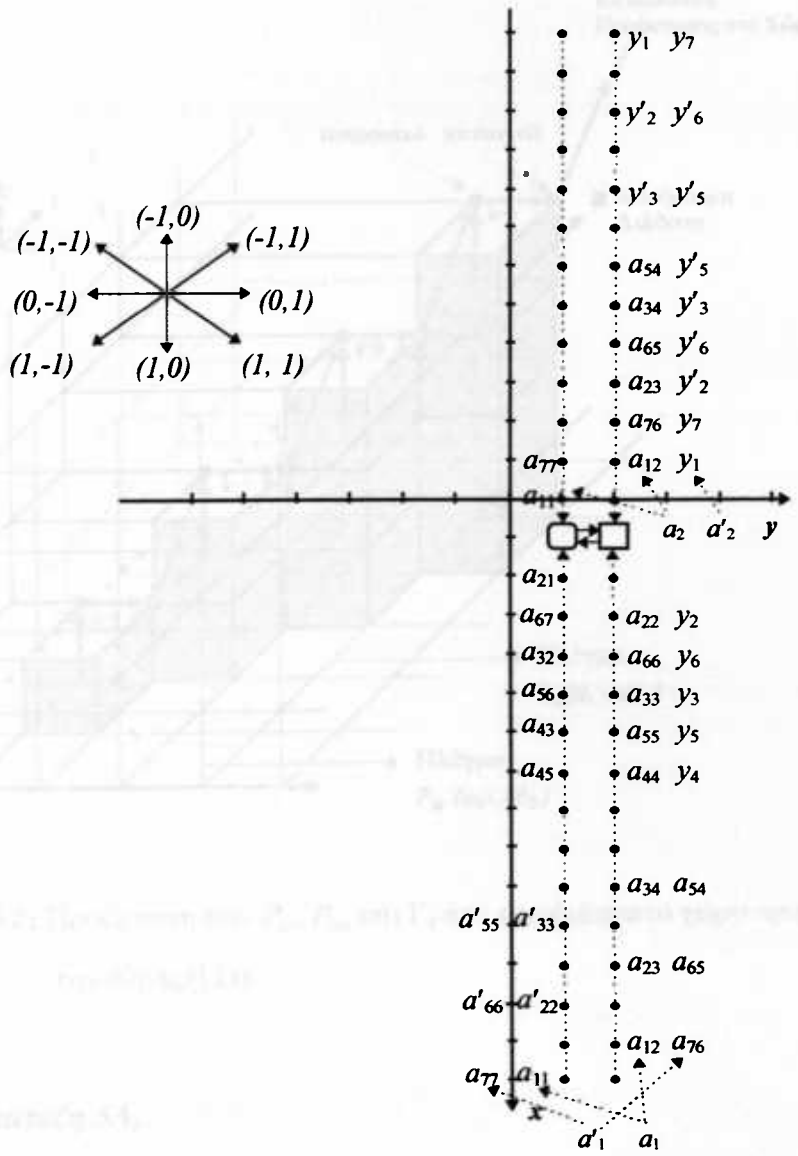
6.2.3.3 Κατεύθυνση Προβολής $\pi_\rho(1,1,1)$

Για την dθρ $\pi_\rho(1,1,1)$, ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$u = u(i) = i,$ $\quad i = 2, ..., n+1$

$v = v(i,j) = i+j-2,$ $\quad i = 2, ..., n+1,\ j = 1, 2$

$w = w(i,k) = i+k-2,$ $\quad i = 2, ..., n+1,\ k = 1$

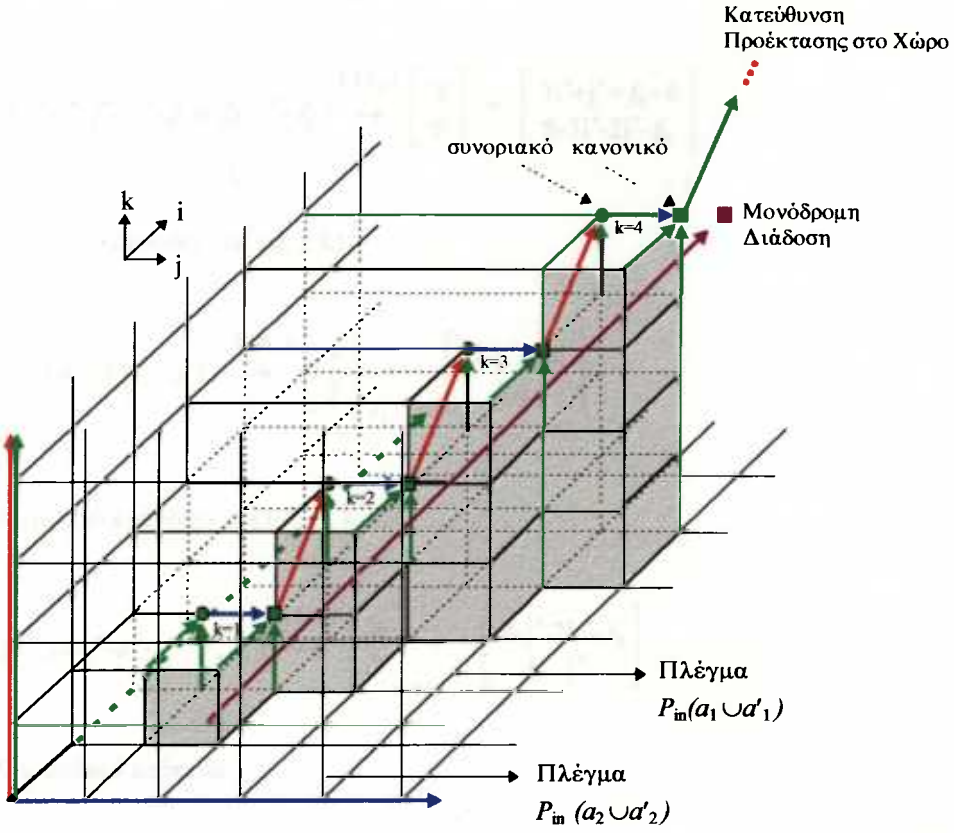


Σχήμα 6.2.3.2-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη (SA_2, d◊p : $\pi_p(1,0,1)$)

Γι' αυτή την d◊p η συστολική διάταξη εκτείνεται στο χώρο, όπως φαίνεται στο σχήμα 6.2.3.3-1.

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων γι' αυτή την d◊p προκύπτουν από τη σχέση (6.2.2.1:1), όπου $f_t = 1,0,-1,\ldots$, $f_b = 2,1,0,-1,\ldots$.

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις :



Σχήμα 6.2.3.3-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στο τρισδιάστατο χώρο προς την $d\Diamond p\ \pi_\rho(1,1,1)$

Συστολική Διάταξη, SA_3:

- $d\Diamond p\ \pi_\rho(1,1,1)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix}$

PE :

$$p(i, i+j-2, i+k-2) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2-k \\ k-j \end{bmatrix}$$

στοιχείο εισόδου a_1 (i, i+j-2, 0):

$$\breve{p}_{\alpha_1}(i, i+j-2, 6-2i-j-f_t) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1} = \begin{bmatrix} 3i+j+f_t-6 \\ 8-3i-2j-f_t \end{bmatrix}$$

στοιχείο εισόδου a'_1 (i^*, i^*+j^*-2, -1):

$$\check{p}_{\alpha'_1}(i^*, i^*+j^*-2, 6-2i^*-j^*-f_b) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_1} = \begin{bmatrix} 3i^*+j^*+f_b-6 \\ 8-3i^*-2j^*-f_b \end{bmatrix}$$

στοιχείο εισόδου a_2 (0, j, k):

$$\check{p}_{\alpha_2}(4-j-k-f_t, j, k) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2} = \begin{bmatrix} 4-j-2k-f_t \\ k-j \end{bmatrix}$$

στοιχείο εισόδου a'_2 (-1, j^*, k^*):

$$\check{p}_{\alpha'_2}(4-j^*-k^*-f_b, j^*, k^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_2} = \begin{bmatrix} 4-j^*-2k^*-f_b \\ k^*-j^* \end{bmatrix}$$

- ***Προβολικό επίπεδο***

Για την dФp π_ρ(1,1,1), η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.2.3.3-2.

6.2.3.4 Κατεύθυνση προβολής π_ρ(0,1,1)

Για την dФp π_ρ(0,1,1), οι υποχώροι P_{in}, P_{int} και P_{out} πρέπει να αναμορφωθούν, γιατί οι ευθείες οι οποίες ορίζονται από τις αντίστοιχες παραμετρικές εξισώσεις διέρχονται από διαφορετικά σημεία του χώρου. Έτσι,

- Για το χώρο P_{in}:

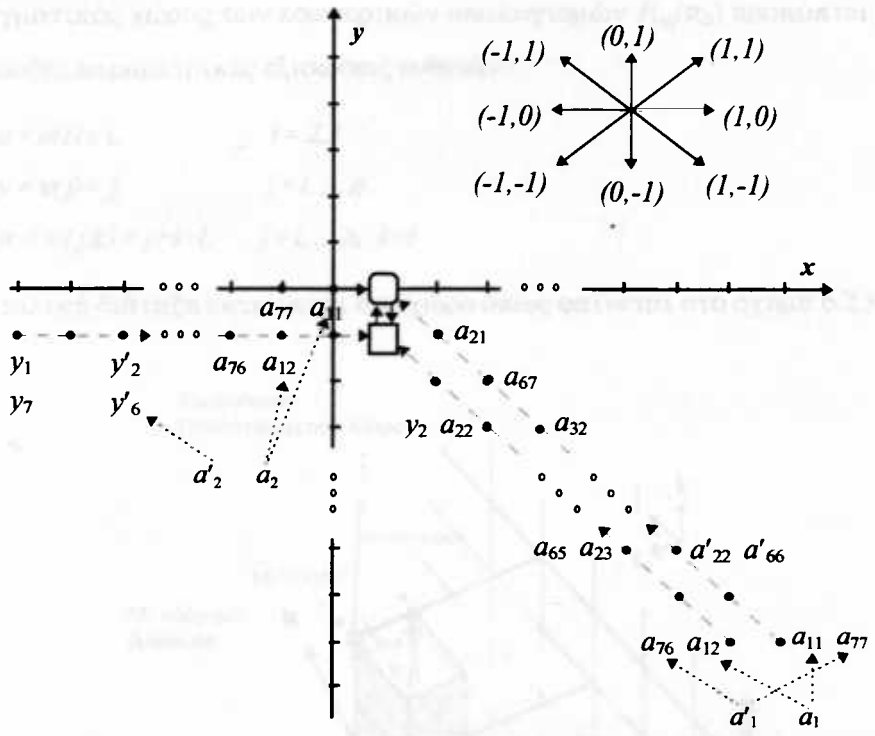
Πάνω ρεύμα

$P_{in}(a_1 \cup a'_1) = \{(i, j, 0) \mid 2 \le i \le 3, 1 \le j \le r\}$, *(el, rhs)*

$P_{in}(a_1) = \{(i, j^*, 0) \mid 2 \le i \le 3, r < j^* \le n\}$, *(bs)*

$P_{in}(a_2 \cup a'_2) = \{(i, 0, k) \mid 2 \le i \le 3, 1 \le k < r\}$, *(el, rhs)*

$P_{in}(a_2) = \{(i, 0, k^*) \mid 2 \le i \le 3, r \le k^* < n\}$, *(bs)*



Σχήμα 6.2.3.3-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

$(SA_3, d\Diamond p : \pi_\rho(1,1,1))$

Κάτω ρεύμα

$P_{in}(a_1 \cup a'_1) = \{(i, j^*, -1) \mid 2 \le i \le 3, 1 \le j^* \le r\}$, *(el, rhs)*

$P_{in}(a'_1) = \{(i, j, 0) \mid 2 \le i \le 3, r < j \le n\}$, *(bs)*

$P_{in}(a_2 \cup a'_2) = \{(i, -1, k^*) \mid 2 \le i \le 3, 1 \le k^* < r\}$, *(el, rhs)*

$P_{in}(a'_2) = \{(i, 0, k) \mid 2 \le i \le 3, r \le k < n\}$, *(bs)*

- Για το χώρο P_{int}:

  $P_{int} = \{(i, j, k) \mid 2 \le i \le 3, 1 \le j \le n, k = 1\}$,

- Για το χώρο P_{out}:

  $P_{out} = \{(i, n+1, k) \mid i = 2, r \le k \le n\}$.

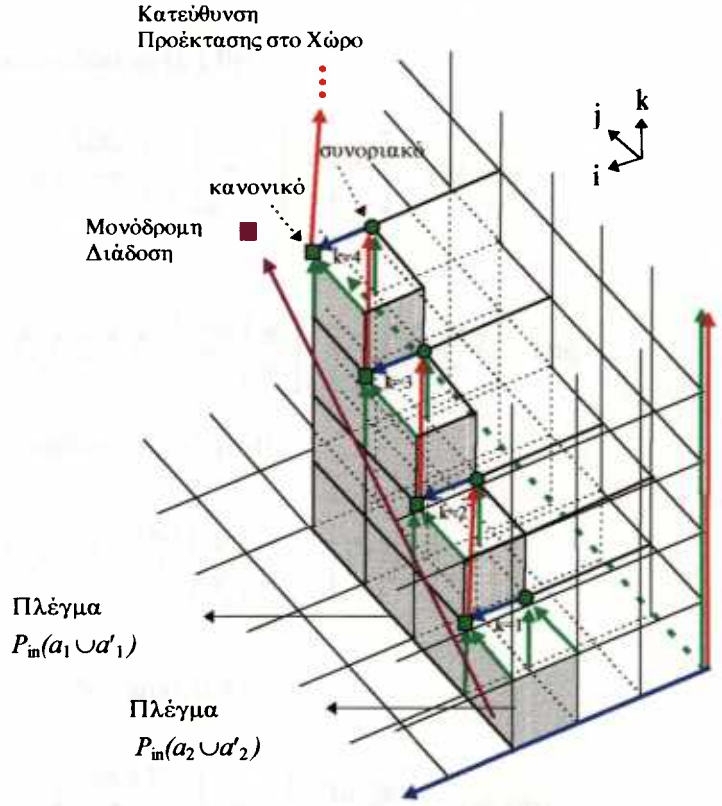
Ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$u = u(i) = i, \qquad i = 2,3$

$v = v(j) = j, \qquad j = 1, ..., n$

$w = w(j,k) = j+k-1, \qquad j = 1, ..., n, \; k=1$

Η συστολική διάταξη εκτείνεται στο χώρο όπως φαίνεται στο σχήμα 6.2.3.4-1



Σχήμα 6.2.3.4-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την $d\Diamond p\, \pi_\rho(0, 1, 1)$

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την $d\Diamond p$ προκύπτουν με βάση τη σχέση (6.2.2.1:1), όπου $f_{t_{el/rhs}} = 1$, $f_{t_{bs}} = 2$, $f_b = 2$.

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_4 :

- dΟp $\pi_\rho(0,1,1)$

- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}$.

PE :

$$p(i, j, j+k-1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-k \\ i \end{bmatrix}$$

στοιχείο εισόδου a_1 (i, j, 0):

$$\check{p}_{\alpha_1}(i, j, 3-i-j) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1} = \begin{bmatrix} i+2j-3 \\ i \end{bmatrix} \quad (el, rhs)$$

or

$$\check{p}_{\alpha_1 \,\&\, \alpha'_1}(i^{\#}, j^{\#}, 2-i^{\#}-j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \,\&\, \alpha'_1} = \begin{bmatrix} i^{\#}+2j^{\#}-2 \\ i^{\#} \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_1 (i^*, j^*, -1):

$$\check{p}_{\alpha'_1}(i^*, j^*, 2-i^*-j^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_1} = \begin{bmatrix} i^*+2j^*-2 \\ i^* \end{bmatrix}$$

στοιχείο εισόδου a_2 (i, 0, k):

$$\check{p}_{\alpha_2}(i, 3-i-k, k) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2} = \begin{bmatrix} 3-i-2k \\ i \end{bmatrix} \quad (el, rhs)$$

or

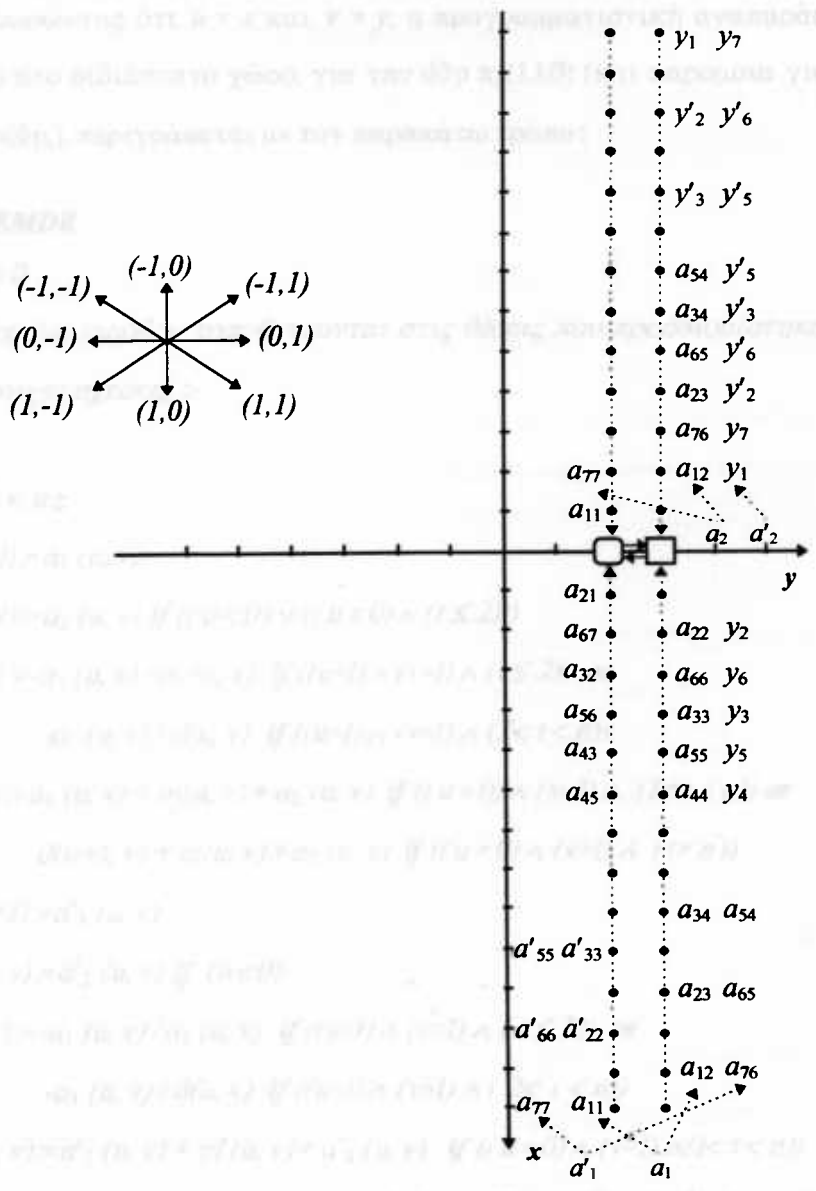
$$\check{p}_{\alpha_2 \,\&\, \alpha'_2}(i^{\#}, 2-i^{\#}-k^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \,\&\, \alpha'_2} = \begin{bmatrix} 2-i^{\#}-2k^{\#} \\ i^{\#} \end{bmatrix} \quad (bs)$$

στοιχείο εισόδου a'_2 (i^*, -1, k^*):

$$\check{p}_{\alpha'_2}(i^*, 2-i^*-k^*, k^*) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha'_2} = \begin{bmatrix} 2-i^*-2k^* \\ i^* \end{bmatrix}$$

- ***Προβολικό επίπεδο***

Για την d◊p $\pi_\rho(0,1,1)$, η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.2.3.4-2.



Σχήμα 6.2.3.4-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

$(SA_4, d◊p : \pi_\rho(0,1,1))$

6.2.4 Προγραμματιστική Αναπαράσταση στο Επίπεδο Προβολής

Η *εντολή πολλαπλής εκχώρησης* η οποία συζητήθηκε αναλυτικά στο Κεφάλαιο 5, χρησιμοποιείται προκειμένου να αποδοθεί ο συγχρονισμός (synchronization) που λαμβάνει χώρα κατά τη διάρκεια της υλοποίησης του αλγόριθμου στη συστολική διάταξη. Θεωρώντας ότι $u = x$ και $v = y$, η προγραμματιστική αναπαράσταση του αλγόριθμου στο διδιάστατο χώρο, για την dôp $\pi_\rho(1,1,0)$ (και παρόμοια για όλες τις επιτρεπτές dôp$_s$), περιγράφεται με τον παρακάτω τρόπο:

Program AEMDR

initially : $t:=0$

< Τα δεδομένα εισόδου τοποθετούνται στις θέσεις που προσδιορίστηκαν από τις αντίστοιχες σχέσεις >

assign :

*< **for all** v, u ::*

|| $a_1(u, v+1) := a_1(u, v)$

|| $a_2(u+1, v) := a_2(u, v)$ ***if*** $((u<0) \vee ((u=0) \wedge (t \leq 2)))$

|| $m(u-1, v) := -a_1(u, v) / a_2(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t \leq 2))$ ***or***

$-a_1(u, v) / d(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (2< t< n))$

|| $d(u+1, v) := a_1(u, v) + m(u, v) * a_2(u, v)$ ***if*** $((u=0) \wedge (v=1) \wedge (1< t< n))$ ***or***

$d(u+1, v) + m(u, v) * a_2(u, v)$ ***if*** $((u=0) \wedge (v=1) \wedge (t=n))$

|| $a'_1(u, v+1) := a'_1(u, v)$

|| $a'_2(u+1, v) := a'_2(u, v)$ ***if*** $(u<0)$

|| $m'(u-1, v) := -a_1(u, v) / a_2(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t \leq 2))$ ***or***

$-a_1(u, v) / d(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (2< t < n))$

|| $y_{mod}(u-1, v) := a'_1(u, v) + m'(u, v) * a'_2(u, v)$ ***if*** $((u=0) \wedge (v=1) \wedge (1< t< n))$

|| $y_{mod}(u+1, v) := y_{mod}(u-1, v) + m'(u, v) * a'_2(u, v)$ ***if*** $((u=0) \wedge (v=1) \wedge (t=n))$

|| $x(u-1, v) := y_{mod}(u, v) / d(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t=n+1))$ ***or***

$s(u, v) / a_1(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t>n+1))$

|| $x'(u-1, v) := y_{mod}(u, v) / d(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t=n+1))$ ***or***

$s'(u, v) / a'_1(u, v)$ ***if*** $((u=1) \wedge (v=1) \wedge (t>n+1))$

|| $s(u+1, v) := a_2(u, v) - a_1(u, v) * x(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t > n+1))$

|| $s'(u+1, v) := a'_2(u, v) - a'_1(u, v) * x'(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t > n+1))$

|| $x(u+1, v) := y_{mod}(u, v) / d(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t = n+1))$ **or**

$s(u, v) / a_1(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t > n+1))$

|| $x'(u+1, v) := s'(u, v) / a'_1(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t > n+1))$ >

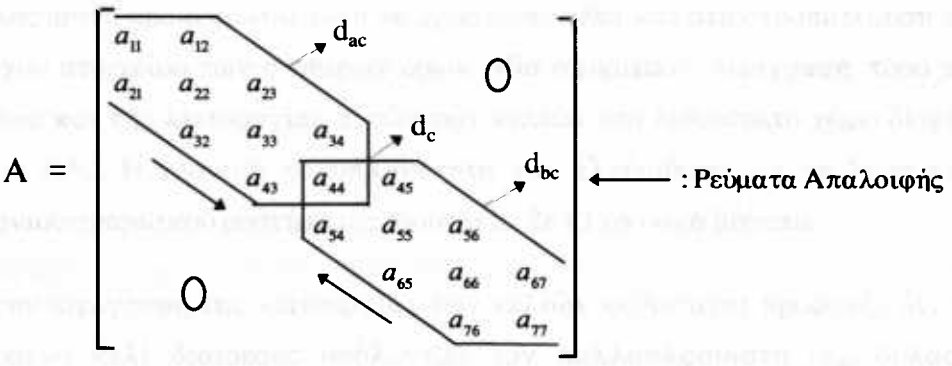
|| $t := t + 1$

end. { AEMDR }

6.3 Χωροαποδοτικός Αλγόριθμος προς τα Πίσω και Εμπρός Απαλοιφής και Επίλυσης (Area Efficient B&F - AEB&F)

Προκειμένου για την αναπαράσταση του αλγόριθμου AEB&F, θεωρούμε και πάλι χωρίς απώλεια της γενικότητας το γραμμικό, συμμετρικό και τριδιαγώνιο σύστημα (6.2:1) για την περίπτωση που το μέγεθος του συντελεστή πίνακα είναι περιττό.

Η προσέγγιση AEB&F υιοθετεί και αυτή την τεχνική R&F κι έτσι η απαλοιφή του πάνω και κάτω ρεύματος δεδομένων ξεκινάει, με τα αντίστοιχα στοιχεία των δύο ρευμάτων δεδομένων να υφίστανται ταυτόχρονη επεξεργασία στα κελιά, κατά τη διάρκεια των ίδιων χρονικών βημάτων. Η μαθηματική μορφοποίηση των δύο διαφορετικών ρευμάτων δεδομένων απεικονίζεται στο σχήμα 6.3-1.



Σχήμα 6.3-1 : Μαθηματική Μορφοποίηση του Πάνω και Κάτω Ρεύματος Δεδομένων (για n=7)

Η πρωτοποριακή ιδέα της συγκεκριμένης προσέγγισης έγκειται στο ότι η παράλληλη απαλοιφή των δύο ρευμάτων δεδομένων δε σταματάει μόλις αυτά συναντηθούν στην κεντρική γραμμή (όταν το μέγεθος του πίνακα είναι περιττός αριθμός) του πίνακα Α, αλλά η διαδικασία συνεχίζει να εφαρμόζεται μέχρις ότου εξαντληθούν όλα τα στοιχεία της υποδιαγωνίου και υπερδιαγωνίου, αντίστοιχα. Το γεγονός αυτό συνεπάγεται τη διπλή τροποποίηση, μία φορά με βάση το πάνω ρεύμα και μία φορά με βάση το κάτω ρεύμα, τόσο κάθε στοιχείου της κεντρικής διαγωνίου, όσο και κάθε στοιχείου του διανύσματος των σταθερών όρων.

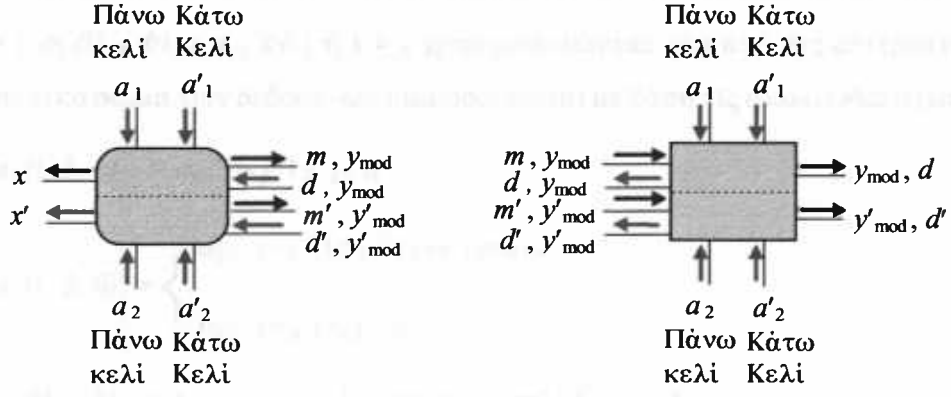
Παρόλα αυτά, η φάση της επίλυσης ξεκινάει αμέσως μετά τη διπλή τροποποίηση του κεντρικού διαγώνιου στοιχείου του πίνακα και εξελίσσεται διαιρώντας διαδοχικά τα στοιχεία του διανύσματος των σταθερών όρων με τα αντίστοιχα στοιχεία της κεντρικής διαγωνίου.

Η συστολική διάταξη που χρησιμοποιεί ο αλγόριθμος AEB&F είναι παρόμοια με αυτή του αλγόριθμου AEMDR και αποτελείται από ένα ‹ 2-σε-1 › εξειδικευμένο κελί διαίρεσης και ένα ‹ 2-σε-1 › κανονικό κελί. Τα κελιά αυτά επιτρέπουν την ταυτόχρονη εκτέλεση δύο IPS πράξεων και αντιστρέφουν τη ροή των δεδομένων, δηλαδή, προωθούν τα δεδομένα το καθένα προς το γειτονικό του κελί, δημιουργώντας έτσι μία εναλλασσόμενης κατεύθυνσης ροή δεδομένων. Η μόνη διαφορά των κελιών αυτών, σε σχέση με τα αντίστοιχα κελιά που χρησιμοποιεί η συστολική διάταξη του αλγόριθμου AEMDR, είναι ότι το ‹ 2-σε-1 › κανονικό κελί διαθέτει έναν καταχωρητή ο οποίος αποθηκεύει την τιμή του εκάστοτε πολλαπλασιαστή προκειμένου αυτή να χρησιμοποιηθεί και στην τροποποίηση του αντίστοιχου στοιχείου των σταθερών όρων. Μία αφηρημένη περιγραφή, τόσο της μορφής, όσο και της λειτουργίας αυτών των κελιών στο διδιάστατο χώρο δίνεται στο σχήμα 6.3-2. Η χρονική πολυπλοκότητα του αλγόριθμου για τη λύση ενός τριδιαγώνιου γραμμικού συστήματος ισούται με 2n+3 χρονικά βήματα.

Από την περιγραφή της λειτουργίας των κελιών καθίσταται προφανές ότι το εξειδικευμένο κελί διαίρεσης υπολογίζει τον πολλαπλασιαστή m_{ik}, δηλαδή, εκτελεί την πράξη $m_{ik} = - a_{ij} / a_{kj}^{(r)}$, όπου i>k ή i<k και k = j. Επιπλέον, μετά από το (n+2)-οστό χρονικό βήμα το κελί αυτό υπολογίζει το διάνυσμα λύσης, δηλαδή., εκτελεί και την πράξη $x_i = y_i'' / a''_{ij}$. Από την άλλη μεριά, το κανονικό κελί τροποποιεί είτε ένα στοιχείο της κύριας διαγωνίου, δηλαδή., εκτελεί την πράξη

$a_{ij}^{(r)} = a_{ij}^{(r)} + m_{ik} * a_{kj}^{(r)}$, όπου i = j και k< j ή k>j, ή ένα στοιχείο του διανύσματος των σταθερών όρων, δηλαδή., εκτελεί την πράξη $y_i^{(r)} = y_i^{(r)} + m_{ik} * y_k^{(r)}$.

Θα πρέπει να σημειωθεί ότι πριν από τον υπολογισμό του διανύσματος λύσης λαμβάνει χώρα μία ενδιάμεση φάση αναδιαμόρφωσης του ρεύματος των δεδομένων. Κατά τη διάρκεια της φάσης αυτής αποδίδονται συγκεκριμένες τιμές στις παραμέτρους p_i και p_{ij} με βάση τις τροποποιημένες τιμές των a_{ij}'s και y_i's.



$m_{ik} = - a_{ik} / a_{kk}^{(r)}$ or $- a_{ik} / d$

or $x_i = y_i'' / a''_{ij}$

(α) ‹2-σε-1› εξειδικευμένο κελί διαίρεσης (*συνοριακό*)

$d = a_{ij}^{(r)} - a_{ij}^{(r)} + m_{ik} * a_{kj}^{(r)}$

or $y_{mod} = y_i^{(r)} = y_i^{(r)} + m_{ik} * y_k^{(r)}$

(β) ‹2-σε-1› κανονικό IPS κελί

ενεργή γραμμή επικοινωνίας : ⟶

Σχήμα 6.3-2: Περιγραφή Λειτουργίας Συστολικών Κελιών

6.3.1 Μονόδρομη Αναπαράσταση στο Χώρο

Για την υλοποίηση του αλγόριθμου, ο υποχώρος, $P \subset Z^3$, θεωρείται ότι αποτελείται από τους ακόλουθους υποχώρους:

- Τον υποχώρο στον οποίο τοποθετείται το *αρχικό ρεύμα των δεδομένων* P_{in}.
- Τον υποχώρο των *εσωτερικών υπολογισμών* P_{int}, και
- Τον υποχώρο των *αποτελεσμάτων εξόδου* P_{out}.

- Το *αρχικό ρεύμα των δεδομένων* τοποθετείται στον υποχώρο $P_{in} = P_{in}(a_1 \cup a'_1) \cup P_{in}(a_2 \cup a'_2)$.

Πάνω Ρεύμα

Για την υλοποίηση της αλγοριθμικής φάσης της *απαλοιφής*, το πάνω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a_1) = \{(i, j, 0) \mid 1 < i \le n, 1 \le j \le 2\}$, δηλαδή, στο επίπεδο k=0 (i,j>0), και $P_{in}(a_2) = \{(0, j, k) \mid 1 \le j \le n, 1 \le k < n\}$, δηλαδή, στο επίπεδο i=0 (j,k>0), όπου πραγματοποιούνται, αντίστοιχα, οι εκχωρήσεις $a_1(i, j, 0) := a_{ij}$, $i > j$ ή $i = j$, $a_2(0, j, k) := a_{kj}$, $k < j$ ή $k = j$, χρησιμοποιώντας μία από τις επιτρεπτές $d0p_s$. Το αρχικό ρεύμα των δεδομένων διαμορφώνεται με βάση τις ακόλουθες σχέσεις:

$a_1(i, 1, 0) := a_{ij}$, $i > j$, $1 \le j < n$

$$a_1(i, 2, 0) := \begin{cases} a_{ij}, & i = j, \ (1 < j \le r) \ \textbf{or} \ (j = n) \\ p_{ij}, & i = j, \ r < j < n \end{cases}$$

$a_2(0, j, k) := a_{kj}$, $k = j = 1$ { *συνοριακό κελί* }

$$a_2(0, j, k) := \begin{cases} a_{kj}, & k < j, \ 1 < j \le r \ \{ \textit{κανονικό κελί} \} \\ 0, & \text{διαφορετικά } \{ \textit{κανονικό κελί} \} \end{cases}$$

όπου $r = \left\lceil \frac{n}{2} \right\rceil$.

Για την υλοποίηση της αλγοριθμικής φάσης της *τροποποίησης του διανύσματος των σταθερών όρων*, το πάνω ρεύμα των δεδομένων τοποθετείται στον υποχώρο $P_{in}(a_1) = \{(i, j, -1) \mid 1 < i \le n, j = 2\}$, δηλαδή, στη γραμμή $j = 2$ του επιπέδου $k = -1$, και $P_{in}(a_2) = \{(-1, j, k) \mid j = 2, k = 1\}$, ενώ οι αντίστοιχες εκχωρήσεις οι οποίες προσδιορίζονται, επίσης, από το αρχικό πάνω ρεύμα δεδομένων είναι:

$$a_1(i, 2, -1) := \begin{cases} y_i, & (1 < i \le r) \ \textbf{or} \ (i = n) \\ p_i, & r < i < n \end{cases}$$

$a_2(-1, k+1, k) := y_k$, $k = 1$ { *κανονικό κελί* }

Κάτω Ρεύμα

Για την υλοποίηση της αλγοριθμικής φάσης της *απαλοιφής*, το κάτω ρεύμα των δεδομένων τοποθετείται στους υποχώρους $P_{in}(a'_1) = \{(i^*, j, 0) \mid 1 < i^* \le n,\ 1 \le j \le 2\}$, δηλαδή, στο επίπεδο k=0 (i,j>0), και $P_{in}(a'_2) = \{(0, j^*, k^*) \mid 1 \le j^* \le n,\ 1 \le k^* < n\}$, δηλαδή, στο επίπεδο i=0 (j,k>0), όπου πραγματοποιούνται, αντίστοιχα, οι εκχωρήσεις $a_1(i^*, j, 0) := a_{ij}$, $i < j$ ή $i = j$, $a_2(0, j^*, k^*) := a_{kj}$ $k > j$ ή $k = j$, χρησιμοποιώντας μία από τις επιτρεπτές $d\Diamond p_s$. Το αρχικό ρεύμα των δεδομένων διαμορφώνεται με βάση τις ακόλουθες σχέσεις:

$a'_1(i^*, 1, 0) := a_{ij}$, $i < j$, $1 < j \le n$

$$a'_1(i^*, 2, 0) := \begin{cases} a_{ij}, & i = j,\ (r < j < n)\ \textbf{or}\ (j = 1) \\ p_{ij}, & i = j,\ 1 < j < r \end{cases}$$

$a'_2(0, j^*, k^*) := a_{kj}$, $k = j = n$ { *συνοριακό κελί* }

$$a'_2(0, j^*, k^*) := \begin{cases} a_{kj}, & k > j,\ \le j < n \quad \{\ \textit{κανονικό κελί}\ \} \\ 0, & \text{διαφορετικά } \{\ \textit{κανονικό κελί}\ \} \end{cases}$$

Για την υλοποίηση της αλγοριθμικής φάσης της *τροποποίησης του διανύσματος των σταθερών όρων*, το κάτω ρεύμα των δεδομένων τοποθετείται στον υποχώρο $P_{in}(a'_1) = \{(i^*, j^*, -1) \mid 1 < i^* \le n,\ j^* = 2\}$, δηλαδή, στη γραμμή j=2 του επιπέδου k=-1, και $P_{in}(a'_2) = \{(-1, j^*, k^*) \mid j^* = 2,\ k^* = n\}$, ενώ οι αντίστοιχες εκχωρήσεις οι οποίες προσδιορίζονται, επίσης, από το αρχικό κάτω ρεύμα δεδομένων είναι:

$$a'_1(i^*, 2, -1) := \begin{cases} y_i, & (r < i < n)\ \textbf{or}\ (i = 1) \\ p_i, & 1 < i < r \end{cases}$$

$a'_2(-1, k^*+1, k^*) := y_k$, $k = n$ { *κανονικό κελί* }

Οι παράμετροι p_i και p_{ij} χρησιμοποιούνται κατά τον ίδιο ακριβώς τρόπο όπως και οι αντίστοιχες παράμετροι στη διαμόρφωση των ρευμάτων δεδομένων του αλγόριθμου AEMDR. Επιπλέον, θα πρέπει να σημειωθεί ότι κατά τη διάρκεια της εκτέλεσης του αλγόριθμου το πάνω και κάτω ρεύμα δεδομένων υφίστανται

επεξεργασία ταυτόχρονα από το πάνω και κάτω τμήμα, αντίστοιχα, του ίδιου κάθε φορά κελιού.

- Οι *εσωτερικοί υπολογισμοί* πραγματοποιούνται στον υποχώρο $P_{int} = \{(i, j, k) \mid 1< i \leq n+1, 1 \leq j \leq 2, k=1\}$. Οι υπολογισμοί που σχετίζονται με κάθε σημείο αυτού του υποχώρου, για το οποίο ισχύει ότι j=1 (συνοριακό κελί), είναι:

$$m_{ik} := -a_{ij} / a_{kj} \textbf{ or } m_{ik} := -a_{ij} / d\{\equiv a'_{kj}\},\ i > k \text{ or } i < k,\ k = j \{\text{πάνω, κάτω κελί}\}$$

δηλαδή,

$$m(i, j, k) := -a_1(i, j, k) / a_2(i, j, k) \textbf{ if } t=1 \textbf{ or}$$

$$-a_1(i, j, k) / d(i, j, k) \textbf{ if } ((t>1) \wedge (t \textbf{ mod } 2 = 1)) \{\text{πάνω κελί}\}$$

Η ίδια πράξη εκτελείται και στο κάτω τμήμα του κελιού χρησιμοποιώντας τις αντίστοιχες μεταβλητές m', a'_1, a'_2 και d'.

Μετά από το (n+2)-οστό χρονικό βήμα οι επιπλέον υπολογισμοί οι οποίοι πραγματοποιούνται στο ίδιο PE, είναι:

$$x_i := y''_i / a''_{ij} \{\text{πάνω, κάτω κελί}\}$$

δηλαδή,

$$x(i, j, k) := y_{mod}(i, j, k) / d(i, j, k) \textbf{ if } ((t > n+2) \wedge (t \textbf{ mod } 2 = 0)) \{\text{πάνω κελί}\}$$

Η ίδια πράξη εκτελείται και στο κάτω τμήμα του κελιού χρησιμοποιώντας τις αντίστοιχες μεταβλητές x', y'_{mod} και d'.

Παρόμοια, οι υπολογισμοί που σχετίζονται με κάθε σημείο αυτού του υποχώρου, για το οποίο ισχύει ότι j=2 (κανονικό κελί), είναι:

$$d \equiv a^{(t)}_{ij} := a^{(t)}_{ij} + m_{ik} * a^{(t)}_{kj},\ i = j,\ k < j \textbf{ or } k > j \{\text{πάνω,κάτω κελί}\}$$

$$y_{mod} \equiv y^{(t)}_i = y^{(t)}_i + m_{ik} * y^{(t)}_k,\ i > k \textbf{ or } i < k \{\text{πάνω,κάτω κελί}\}$$

δηλαδή,

$$d(i, j, k) := a_1(i, j, k) + m(i, j, k) * a_2(i, j, k) \textbf{ if } ((t > 1) \wedge (t \textbf{ mod } 2 = 0))$$

$$\{\text{πάνω κελί}\}, \text{ και}$$

$$y_{mod}(i, j, k) := a_1(i, j, k) + m(i, j, k) * a_2(i, j, k) \textbf{ if } (t = 3) \textbf{ or}$$

$$a_1(i, j, k) + m(i, j, k) * y_{mod}(i, j, k) \textbf{ if } ((t > 3) \wedge (t \textbf{ mod } 2 = 1))$$

$$\{\text{πάνω κελί}\}$$

Η ίδια πράξη εκτελείται και στο κάτω τμήμα του κελιού χρησιμοποιώντας τις αντίστοιχες μεταβλητές m', a'_1, a'_2, d' και y'_{mod}.

- Τα *αποτελέσματα εξόδου* τοποθετούνται στον υποχώρο $P_{out} = P_{out}(x) \cup P_{out}(a') \cup P_{out}(y')$.

Ο υποχώρος $P_{out}(x)$ ορίζεται ως $P_{out}(x) = \{(n+2, j, k), r \le j, k \le n\}$. Οι εκχωρήσεις που πραγματοποιούνται, οι οποίες αντιστοιχούν στο διάνυσμα λύσης του συστήματος, $\vec{x}$, είναι:

$$x_i = \left\{ \begin{array}{l} x(n+2, j, k) := x(n+2-m, j, k),\ i = m \ \{\textit{πάνω συνοριακό κελί}\} \\ x'(n+2, j, k) := x'(n+2-m, j, k),\ i = n-m+1 \{\textit{κάτωσυνοριακό κελί}\} \end{array} \right\}$$

για $i=1,\ldots,n$, $m = r, r-1,\ldots,1$, και $t = n + 2p$, για $1 < p \le r+1$, όταν το n είναι περιττός αριθμός.

Ο υποχώρος $P_{out}(a')$ ορίζεται ως $P_{out}(a') = \{(i, n+1, k), 1 < i \le r, 1 \le k < r\}$. Οι εκχωρήσεις που σχετίζονται με κάθε σημείο αυτού του υποχώρου είναι:

$$a'_{jj} = \left\{ \begin{array}{l} d(i, n+1, k) := d(i, n+1-m, k),\ j = n-m+1 \{\textit{πάνω συνοριακό κελί}\} \\ d'(i, n+1, k) := d'(i, n+1-m, k),\ j = m \{\textit{κάτω συνοριακό κελί}\} \end{array} \right\}$$

για $j=2,\ldots,n-1$, $m = n-1, n-2,\ldots, r$, και $t = 1 + 2p$, για $1 \le p < r$, όταν το n είναι περιττός αριθμός.

Ο υποχώρος $P_{out}(y')$ ορίζεται ως $P_{out}(y') = \{(i, n+2, k), 1 < i \le r, 1 \le k < r\}$. Οι εκχωρήσεις που σχετίζονται με κάθε σημείο αυτού του υποχώρου είναι:

$$y'_j = \left\{ \begin{array}{l} y_{mod}(i, n+2, k) := y_{mod}(i, n+2-m, k),\ j = n-m+2 \{\textit{πάνω συνοριακό κελί}\} \\ y'_{mod}(i, n+2, k) := y'_{mod}(i, n+2-m, k),\ j = m-1 \{\textit{κάτω συνοριακό κελί}\} \end{array} \right\}$$

για $j=2, \ldots, n-1$, $m = n, n-1,\ldots, r+1$, και $t = 2 + 2p$, για $1 \le p < r$, όταν το n είναι περιττός αριθμός.

Και σ' αυτή την περίπτωση, η αναπαράσταση των υπολογισμών στο χώρο είναι εφικτή μόνον για τις $d\Diamond p_s$ (1,1,0), (1,0,1), (0,1,1) και (1,1,1). Οι $d\Diamond p_s$ (1,0,0), (0,1,0) και (0,0,1) δεν αποτελούν επιτρεπτές κατευθύνσεις προβολής, γιατί σε οποιαδήποτε από αυτές ένα τμήμα του χώρου P_{int} επικαλύπτεται από ένα τμήμα του χώρου P_{in}.

Κατά συνέπεια, ο χώρος P_{in} $(a_2 \cup a'_2)$ βρίσκεται στο επίπεδο i=0 (j,k >0) για τις $d\Diamond p_s$ (1,1,0), (1,0,1) και (1,1,1). Ο χώρος αυτός στον οποίο τοποθετούνται τα αρχικά

δεδομένα είναι η γραμμή k=1, του επιπέδου i=0, για την dΟp (1,1,0), οι γραμμές j=1 και j=2, του επιπέδου i=0, για την dΟp (1,0,1), και το επίπεδο i=0 (j,k >0), για την dΟp (1,1,1). Όσον αφορά στην dΟp (0,1,1), τα στοιχεία της μήτρας a_{kj}, για k<j ή k>j, y_k ή y_i, εισέρχονται στη συστολική διάταξη από το επίπεδο j = 0 (i,k >0), αντί από το επίπεδο i = 0 (j,k>0), όπως συμβαίνει στις τρεις προηγούμενες dΟp$_s$.

Επιπλέον, υπενθυμίζεται ότι η εφαρμογή της τεχνικής R&F, επιβάλλει την επέκταση του πρώτου οκτάεδρου του χώρου, στο οποίο θεωρούμε ότι υλοποιείται ο αλγόριθμος, έτσι ώστε να συμπεριληφθούν και τα επίπεδα i = -1 (j,k >0), j = -1 (i,k>0) και k = -1 (i,j>0), στα οποία όπως ήδη αναφέρθηκε τοποθετείται το αρχικό κάτω ρεύμα δεδομένων για τις αλγοριθμικές φάσεις της απαλοιφής και τροποποίησης του διανύσματος των σταθερών όρων.

6.3.2 Προγραμματιστική Αναπαράσταση στο Χώρο

Στην Παράγραφο αυτή παρουσιάζεται η προγραμματιστική αναπαράσταση του επαναληπτικού συστολικού αλγόριθμου AEB&F, κάνοντας χρήση της συγκεκριμένης κατεύθυνσης προβολής (1,1,0).

Algorithm_AEB&F
t:=1
for *i:=2* ***to*** *n+1* ***do***
 for *j:= i-1* ***to*** *i* ***do***
  if *j< i* ***then*** *{συνοριακό κελί}*
   $a_1(i, j, k) := a_1(i, j, 0)$
   $a'_1(i, j, k) := a'_1(i, j, 0)$
   $a_2(i, j, k) := a_2(0, j, k)$ ***if*** *(t=1)*
   $a'_2(i, j, k) := a'_2(0, j, k)$ ***if*** *(t=1)*
   $d(i, j, k) := d(i-1, j, k)$ ***if*** $((1< t<n) \wedge (t> n) \wedge (t \neq n+2) \wedge (t$ ***mod*** $2 = 1))$
   $d'(i-1, j, k)$ ***if*** $(t = n+1)$
   $d'(i, j, k) := d'(i-1, j, k)$ ***if*** $(((2< t<n) \wedge (t > n+1) \wedge (t \neq n+2) \wedge (t$ ***mod*** $2 = 1)) \vee$
   $(t = n+1))$

$m(i, j, k) := -a_1(i, j, k) / a_2(i, j, k)$ **if** $t = 1$ **or**

$-a_1(i, j, k) / d(i, j, k)$ **if** $(((2 < t < n) \wedge (t > n+1)) \wedge (t \neq n+2) \wedge$

$(t \bmod 2 = 1)) \vee (t = n+1))$

$m'(i, j, k) := -a'_1(i, j, k) / a'_2(i, j, k)$ **if** $t = 1$ **or**

$-a'_1(i, j, k) / d'(i, j, k)$ **if** $(((2 < t < n) \wedge (t > n+1) \wedge (t \neq n+2) \wedge$

$(t \bmod 2 = 1)) \vee (t = n+1))$

$y_{mod}(i, j, k) := y_{mod}(i-1, j, k)$ **if** $((3 < t < n) \wedge (t > n+1) \wedge (t \neq n+3) \wedge (t \bmod 2 = 0))$

$y'_{mod}(i-1, j, k)$ **if** $(t = n+3)$

$y'_{mod}(i, j, k) := y'_{mod}(i-1, j, k)$ **if** $((3 < t < n) \wedge (t > n+1) \wedge (t \bmod 2 = 0))$

$x(i, j, k) := y_{mod}(i, j, k) / d(i, j, k)$ **if** $((t > n+1) \wedge (t \bmod 2 = 0))$

$x'(i, j, k) := y'_{mod}(i, j, k) / d'(i, j, k)$ **if** $((t > n+3) \wedge (t \bmod 2 = 0))$

else if $j = i$ **then** { κανονικό κελί }

$a_1(i, j, k) := a_1(i, j, 0)$

$a'_1(i, j, k) := a'_1(i, j, 0)$ **if** $((t < n-1) \vee (t > n+2))$ **or**

$d(i, j, k)$ **if** $(t = n)$ **or**

$y_{mod}(i, j, k)$ **if** $(t = n+2)$

$a_2(i, j, k) := a_2(0, j, k)$

$a'_2(i, j, k) := a'_2(0, j, k)$

$m(i, j, k) := m(i, j-1, k)$ **if** $((1 < t < n) \wedge (t > n+1) \wedge (t \bmod 2 = 0))$

$m'(i, j, k) := m'(i, j-1, k)$ **if** $((1 < t < n) \wedge (t > n+1) \wedge (t \bmod 2 = 0))$

$d(i, j, k) := a_1(i, j, k) + m(i, j, k) * a_2(i, j, k)$ **if** $((1 < t < n) \wedge (t > n+1) \wedge (t \bmod 2 = 0))$

$d'(i, j, k) := a'_1(i, j, k) + m'(i, j, k) * a'_2(i, j, k)$ **if** $(((1 < t < n-1) \wedge (t > n+1) \wedge$

$(t \bmod 2 = 0)) \vee (t = n))$

$y_{mod}(i, j, k) := y_{mod}(i, j-1, k)$ **if** $((((3 < t < n) \vee (t > n+2)) \wedge (t \bmod 2 = 1)) \vee (t = n+1))$

$y'_{mod}(i, j, k) := y'_{mod}(i, j-1, k)$ **if** $((((3 < t < n) \vee (t > n+2)) \wedge (t \bmod 2 = 1)) \vee$

$(t = n+1))$

$y_{mod}(i, j, k) := a_1(i, j, k) + m(i, j, k) * a_2(i, j, k)$ **if** $(t = 3)$ **or**

$a_1(i, j, k) + m(i, j, k) * y_{mod}(i, j, k)$ **if** $((((3 < t < n) \vee (t > n+2)) \wedge$

$(t \bmod 2 = 1)) \vee (t = n+1))$

$y'_{mod}(i,j,k) := a'_1(i,j,k) + m'(i,j,k) * a'_2(i,j,k)$ **if** $(t=3)$ **or**

$a'_1(i,j,k) + m'(i,j,k) * y'_{mod}(i,j,k)$ **if** $(((3 < t < n) \vee (t > n)) \wedge$

$(t \bmod 2 = 1))$

$t := t+1$

Η εφαρμογή της τεχνικής R&F, και στη συγκεκριμένη περίπτωση, συνεπάγεται ότι ο συγκεκριμένος αλγόριθμος εφαρμόζεται ταυτόχρονα και στο πάνω, αλλά και στο κάτω ρεύμα δεδομένων, τα οποία διεμπλέκονται καθώς υφίστανται επεξεργασία κατά τη διάρκεια της φάσης της απαλοιφής και της φάσης της τροποποίησης του διανύσματος των σταθερών όρων, με το πάνω ρεύμα να προηγείται κατά ένα χρονικό βήμα. Το γεγονός αυτό συνεπάγεται ότι, όταν τα δεδομένα εισόδου που αντιστοιχούν στο πάνω ρεύμα εισέρχονται στη συστολική διάταξη, τα αντίστοιχα δεδομένα εισόδου του κάτω ρεύματος μετακινούνται από τα επίπεδα k = -1 (i,j>0) και i = -1 (j,k>0), στα επίπεδα k=0 (i,j>0) και i=0 (j,k>0), προκειμένου να εισέλθουν στα PEs κατά το επόμενο χρονικό βήμα.

6.3.2.1 Εξαρτήσεις στο Γράφημα Υπολογιστικής Δραστηριότητας

Το γράφημα καθολικών εξαρτήσεων, για τον αλγόριθμο AEB&F, είναι της μορφής $\Gamma_g = (P_{int}, \vec{e}^3_{\alpha_1}, \vec{Q}^3_2, \vec{e}^3_m, \vec{Q}^3_3, \vec{e}^3_{\alpha'_1}, \vec{Q}'^3_2, \vec{e}^3_{m'}, \vec{Q}'^3_3)$ και περιγράφει τη διάδοση των υπολογισμών στο χώρο P. Τα διανύσματα $\vec{Q}^3_2(i,0,0)$, $\vec{Q}'^3_2(i,0,0)$, $\vec{Q}^3_3(0,j,0)$, $\vec{Q}'^3_3(0,j,0)$ είναι τα *καθολικά* διανύσματα εξαρτήσεων, ενώ τα διανύσματα $\vec{e}^3_{\alpha_1}(0,0,1)$, $\vec{e}^3_{\alpha'_1}(0,0,1)$, $\vec{e}^3_m(0,1,0)$, $\vec{e}^3_{m'}(0,1,0)$ είναι τα *τοπικά* διανύσματα εξαρτήσεων. Τα διανύσματα $\vec{e}^3_{\alpha_1}$, $\vec{e}^3_{\alpha'_1}$, $\vec{Q}^3_2$, $\vec{Q}'^3_2$ εκφράζουν, ουσιαστικά, τη συσχέτιση που υπάρχει μεταξύ του υπολογισμού που πραγματοποιείται στο σημείο $p(u, v, w) \in P_{int}(\pi_\rho)$ και των αντίστοιχων δεδομένων που εκχωρούνται στις κατάλληλες μεταβλητές εισόδου. Τα διανύσματα $\vec{Q}^3_3, \vec{Q}'^3_3$ αφορούν στη συσχέτιση που υπάρχει μεταξύ των αποτελεσμάτων που παράγονται στο σημείο $p(u, v, w) \in P_{int}(\pi_\rho)$ και των αντίστοιχων θέσεων του χώρου $P_{out}(\pi_\rho)$ στις οποίες αυτά εκχωρούνται.

Προκειμένου για την απαλοιφή των καθολικών εξαρτήσεων για τα δεδομένα εισόδου a_{kj}, y_k ή y_i, θεωρούμε ότι

$$\varepsilon_2(p) = a_2(p\text{-}\vec{Q}_2^3) = \begin{cases} a_{kj}, & 1 \le j, k \le n \\ y_i, & 1 \le i < r \end{cases}$$

$$\varepsilon'_2(p) = a'_2(p\text{-}\vec{Q}_2'^3) = \begin{cases} y_k, & 1 \le j, k \le n \\ y_i, & r < i \le n, \end{cases}$$

όπου το $p \in P_{int}(\pi_p)$. Παρόμοια, το διάνυσμα λύσης, $\vec{x}$, λαμβάνεται θεωρώντας ότι

$\varepsilon_3(p) = x(p\text{-}\vec{Q}_2^3) = x_i, 1 \le i \le r$ { *πάνω ρεύμα* } και

$\varepsilon'_3(p) = x'(p\text{-}\vec{Q}_2'^3) = x_i, r < i \le n$ { *κάτω ρεύμα* }, όπου το $p \in P_{out}(x)$,

ενώ αντίστοιχα, τα διαγώνια στοιχεία του συντελεστή πίνακα Α, καθώς επίσης και τα στοιχεία του διανύσματος των σταθερών όρων μετά από την πρώτη τροποποίηση λαμβάνονται θεωρώντας ότι

$\varepsilon_4(p) = d(p\text{-}\vec{Q}_3^3) = a'_{ii}, 1 < i < r$ { *πάνω ρεύμα* } και

$\varepsilon'_4(p) = d'(p\text{-}\vec{Q}_3'^3) = a'_{ii}, r \le i < n$ { *κάτω ρεύμα* }, όπου το $p \in P_{out}(a')$,

και

$\varepsilon_5(p) = y_{mod}[p\text{-}(\vec{Q}_3^3 + \vec{e}_{m'}^3)] = y_i', 1 < i < r$ { *πάνω ρεύμα* } και

$\varepsilon'_5(p) = y'_{mod}[p\text{-}(\vec{Q}_3'^3 + \vec{e}_{m'}^3)] = y_i', r \le i < n$ { *κάτω ρεύμα* }, όπου το $p \in P_{out}(y')$.

Εξαλείφοντας τις καθολικές εξαρτήσεις $\vec{Q}_2^3$, $\vec{Q}_2'^3$, $\vec{Q}_3^3$, $\vec{Q}_3'^3$ το γράφημα Γ_g αντικαθίσταται από ένα ισοδύναμό του γράφημα Γ_ℓ = (P_{int}, $\vec{e}_{\alpha_1}^3$, $\vec{e}_{\alpha_2}^3$, $\vec{e}_m^3$, $\vec{e}_{\alpha_1'}^3$, $\vec{e}_{\alpha_2'}^3$, $\vec{e}_{m'}^3$), όπου τα διανύσματα $\vec{e}_{\alpha_2}^3$ (1,0,0), $\vec{e}_{\alpha_2'}^3$ (1,0,0) είναι τα τοπικά διανύσματα εξάρτησης για τα δεδομένα εισόδου a_{kj}, y_k ή y_i, αντίστοιχα. Το γράφημα που προέκυψε μπορεί, πλέον, να απεικονιστεί σε μια μονοδιάστατη συστολική διάταξη κάνοντας χρήση μιας επιτρεπτής $d\Diamond p$.

Απεικονίζοντας το γράφημα Γ_ℓ κατά μήκος της d0p $\pi_\rho(\pi_i,\pi_j,\pi_k)$, προκύπτει ένας μονοδιάστατος συστολικός πίνακας (1D-SA) μεγέθους w-1. Ωστόσο, οι σωστές θέσεις των δεδομένων εισόδου στο προβολικό επίπεδο παρέχονται μόνον μέσω της επέκτασης του χώρου του αρχικού ρεύματος των δεδομένων, $P_{in}(\pi_\rho)$. Οι νέες θέσεις των αρχικών δεδομένων εισόδου στο χώρο Z^3, λαμβάνονται με βάση την ακόλουθη σχέση, η οποία χρησιμοποιείται για κάθε ξεχωριστή d0p.

$$\check{p}_\varepsilon = position(p_\varepsilon) = \vec{p}_\varepsilon - [t(\check{p}_\varepsilon) + f_g]\,\vec{e}_\varepsilon^{\,3}, \qquad (6.3.2.1:1)$$

$$f_g = \{ f_{el(imination)}, f_{rhs(modification)} \},\ \varepsilon = \{ a_1, a_2, a'_1, a'_2 \}$$

Στο συγκεκριμένο αλγόριθμο έχουμε διάκριση των τιμών της παραμέτρου *f*, όχι μεταξύ των δεδομένων του πάνω και του κάτω ρεύματος, όπως συμβαίνει στο αλγόριθμο AEMDR, αλλά μεταξύ των δεδομένων των φάσεων της απαλοιφής και της τροποποίησης του διανύσματος των σταθερών όρων. Ουσιαστικά θεωρούμε, ότι τα δεδομένα του αλγόριθμου χωρίζονται σε τρία νοητά τμήματα, όπως φαίνεται στο σχήμα 6.3-1 : Στα δεδομένα πάνω από το κεντρο (data above centre - d_{ac}), στα κεντρικά δεδομένα (data of centre - d_c) και στα δεδομένα κάτω από το κέντρο (data below centre - d_{bc}). Όταν λέμε κεντρικά δεδομένα, εννοούμε τα δεδομένα που αντιστοιχούν στην κεντική γραμμή του συντελεστή πίνακα A και του διανύσματος των σταθερών όρων, όταν το n είναι περιττός αριθμός. Επίσης, και το διάνυσμα των σταθερών όρων θεωρείται ότι χωρίζεται σε τρία τμήματα, αντίστοιχα. Με βάση τα παραπάνω, οι τιμές της *f* διαμορφώνονται ανάλογα με τη φάση στην οποία συμμετέχουν, το τμήμα στο οποίο ανήκουν και τον τύπο του κελιού στο οποίο εισέρχονται, τα εκάστοτε δεδομένα εισόδου.

Η συνάρτηση χρόνου ορίζεται ως $t(p_\varepsilon) = i+j+k-4$, εφόσον ισχύει ότι $t(p_{start}) = p(2,1,1)$. Επιπλέον, θα πρέπει να σημειωθεί ότι, η διαδικασία επέκτασης εφαρμόζεται μόνον στο χώρο του αρχικού ρεύματος των δεδομένων και όχι στο χώρο των εσωτερικών υπολογισμών, $P_{int}(\pi_\rho)$.

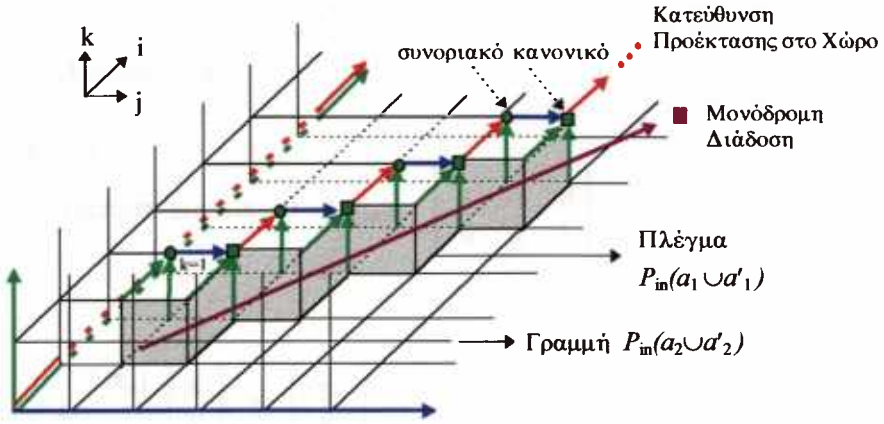
6.3.3 Προβολικά Επίπεδα

6.3.3.1 Κατεύθυνση Προβολής $\pi_\rho(1,1,0)$

Για την d◊p $\pi_\rho(1,1,0)$, ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$$u = u(i) = i, \qquad i = 2, \ldots, n+1$$
$$v = v(i,j) = i+j-2, \qquad i = 2, \ldots, n+1, \; j = 1,2$$
$$w = w(k) = k, \qquad k = 1$$

Η απεικόνιση αυτή εφαρμόζεται, επίσης, και στα σημεία του χώρου P_{in} $(a_1 \cup a'_1)$, αλλά όχι σε αυτά του χώρου P_{in} $(a_2 \cup a'_2)$. Η σύμβαση αυτή ισχύει για όλες τις επιτρεπτές d◊p_s που θα εξεταστούν στη συνέχεια του παρόντος Κεφαλαίου. Για τη συγκεκριμένη d◊p, η συστολική διάταξη εκτείνεται στο επίπεδο k=1 (i,j>0), όπως φαίνεται στο σχήμα 6.3.3.1-1.



Σχήμα 6.3.3.1-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την d◊p $\pi_\rho(1,1,0)$

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την d◊p προκύπτουν από τη σχέση (6.3.2.1:1), στην οποία οι τιμές της f προσδιορίζονται με βάση τον παρακάτω πίνακα:

Αλγοριθμική Φάση	Συστολικά Κελιά	a_1	&	a'_1	a_2	&	a'_2
		d_{ac}	d_c	d_{bc}	d_{ac}	d_c	d_{bc}
Απαλοιφή	Συνοριακό	1	2	3	1	-	-
	Κανονικό	1	-	3	1	-	3
Τροποποίηση	Συνοριακό	-	-	-	-	-	-
Σταθερών Όρων	Κανονικό	2	3	4	2	-	-

Πίνακας 6.3.3.1-1: Οι Τιμές των Παραμέτρων f_{el} και f_{rhs} για τα Διάφορα Τμήματα Δεδομένων για τις $d\Diamond p_s$: $\pi_\rho(1,1,0)$, $\pi_\rho(1,0,1)$ και $\pi_\rho(0,1,1)$

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_1:

- $d\Diamond p\ \pi_\rho(1,1,0)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

PE:

$$p(i, i+j-2, k) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2-j \\ k \end{bmatrix}$$

στοιχεία εισόδου a_1 $(i, i+j-2, k)$, a'_1 (i^*, i^*+j^*-2, k), όπου $k,k^* \in \{0,-1\}$:

$$\check{p}_{\alpha_1 \,\&\, \alpha'_1}\ (i^\#, i^\#+j^\#-2, 5-2i^\#-j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \,\&\, \alpha'_1} = \begin{bmatrix} 2-j^\# \\ 5-2i^\#-j^\# \end{bmatrix},\ (el: d_{ac})$$

$$\check{p}_{\alpha_1 \,\&\, \alpha'_1}\ (i^\#, i^\#+j^\#-2, 4-2i^\#-j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \,\&\, \alpha'_1} = \begin{bmatrix} 2-j^\# \\ 4-2i^\#-j^\# \end{bmatrix},\ (el: d_c,\ r.h.s.: d_{ac})$$

$$\check{p}_{\alpha_1 \& \alpha_1'}\ (i^{\#}, i^{\#}+j^{\#}-2, 3-2i^{\#}-j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha_1'} = \begin{bmatrix} 2-j^{\#} \\ 3-2i^{\#}-j^{\#} \end{bmatrix},\ (el: d_{bc},\ r.h.s.: d_c)$$

$$\check{p}_{\alpha_1 \& \alpha_1'}\ (i^{\#}, i^{\#}+j^{\#}-2, 2-2i^{\#}-j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha_1'} = \begin{bmatrix} 2-j^{\#} \\ 2-2i^{\#}-j^{\#} \end{bmatrix},\ (r.h.s.: d_{bc})$$

στοιχεία εισόδου a_2 (i, j, 1), a'_2 (i^*, j^*, 1), όπου $i,i^* \in \{0,-1\}$:

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (2-j^{\#}, j^{\#}, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 2-2j^{\#} \\ 1 \end{bmatrix},\ (el: d_{ac})$$

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (-j^{\#}, j^{\#}, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} -2j^{\#} \\ 1 \end{bmatrix},\ (el: d_{bc})$$

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (1-j^{\#}, j^{\#}, 1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 1-2j^{\#} \\ 1 \end{bmatrix},\ (r.h.s.: d_{ac})$$

- ***Προβολικό επίπεδο***

Για την dθp π_ρ(1,1,0), η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.3.3.1-2.

6.3.3.2 Κατεύθυνση Προβολής π_ρ(1,0,1)

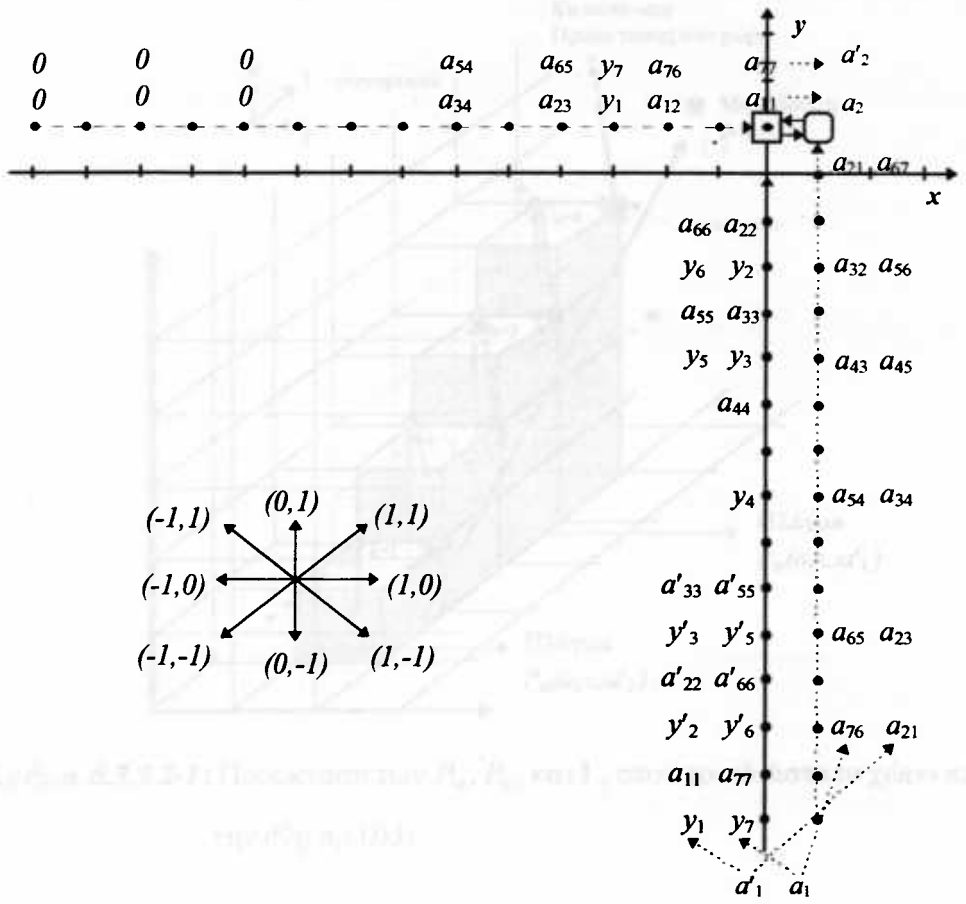
Για την dθp π_ρ(1,0,1), ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών :

$u = u(i) = i,$ $\quad i = 2, ..., n+1$

$v = v(j) = j,$ $\quad j = 1,2$

$w = w(i,k) = i+k-2,$ $\quad i = 2, ..., n+1,\ k = 1$

Γι' αυτή την dθp η συστολική διάταξη εκτείνεται στα επίπεδα j=1 και j=2 (i,k>0), όπως φαίνεται στο σχήμα 6.3.3.2-1.



Σχήμα 6.3.3.1-2 : Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

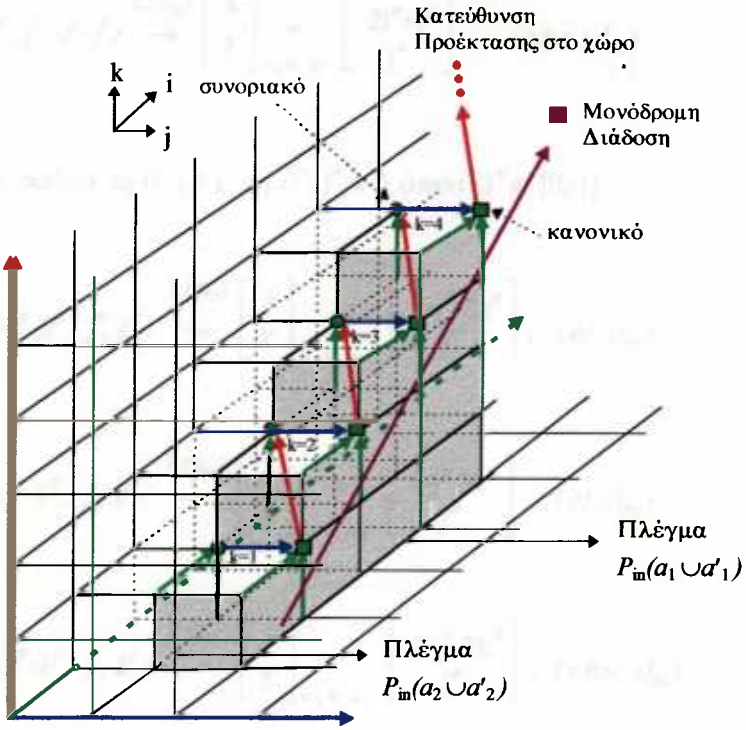
$(SA_1, d\Diamond p : \pi_\rho(1,1,0))$

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την $d\Diamond p$ προκύπτουν από τη σχέση (6.3.2.1:1), ενώ οι παράμετροι f_{el} και f_{rhs} παίρνουν κάθε φορά την κατάλληλη τιμή με βάση τον πίνακα 6.3.3.1-1 .

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις :

Συστολική Διάταξη, SA_2:

- $d\Diamond p\ \pi_\rho(1,0,1)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$



Σχήμα 6.3.3.2-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την d◊p $\pi_\rho(1,0,1)$

PE :

$$p(i, j, i+k-2) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2\text{-}k \\ j \end{bmatrix}$$

στοιχεία εισόδου a_1 (i, j, k), a'_1 (i^*, j^*, k^*), όπου $k,k^* \in \{0,-1\}$:

$$\check{p}_{\alpha_1 \& \alpha'_1}(i^\#, j^\#, 3\text{-}i^\#\text{-}j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 2i^\#+j^\#-3 \\ j^\# \end{bmatrix}, \quad (el : d_{ac})$$

$$\check{p}_{\alpha_1 \& \alpha'_1}(i^\#, j^\#, 2\text{-}i^\#\text{-}j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 2i^\#+j^\#-2 \\ j^\# \end{bmatrix}, \quad (el : d_c, r.h.s. : d_{ac})$$

$$\check{p}_{\alpha_1 \& \alpha'_1}(i^\#, j^\#, 1\text{-}i^\#\text{-}j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 2i^\#+j^\#-1 \\ j^\# \end{bmatrix}, \quad (el : d_{bc}, r.h.s. : d_c)$$

$$\check{p}_{\alpha_1 \& \alpha_1'}\ (i^{\#}, j^{\#}, -i^{\#}-j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha_1'} = \begin{bmatrix} 2i^{\#}+j^{\#} \\ j^{\#} \end{bmatrix}, \quad (r.h.s. : d_{bc})$$

στοιχεία εισόδου a_2 (i, j, k), a'_2 (i^*, j^*, k^*), όπου $i,i^* \in \{0,-1\}$:

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (3-j^{\#}-k^{\#}, j^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 3-j^{\#}-2k^{\#} \\ j^{\#} \end{bmatrix}, \quad (el : d_{ac})$$

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (1-j^{\#}-k^{\#}, j^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 1-j^{\#}-2k^{\#} \\ j^{\#} \end{bmatrix}, \quad (el : d_{bc})$$

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (2-j^{\#}-k^{\#}, j^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 2-j^{\#}-2k^{\#} \\ j^{\#} \end{bmatrix}, \quad (r.h.s. : d_{ac})$$

- ***Προβολικό επίπεδο***

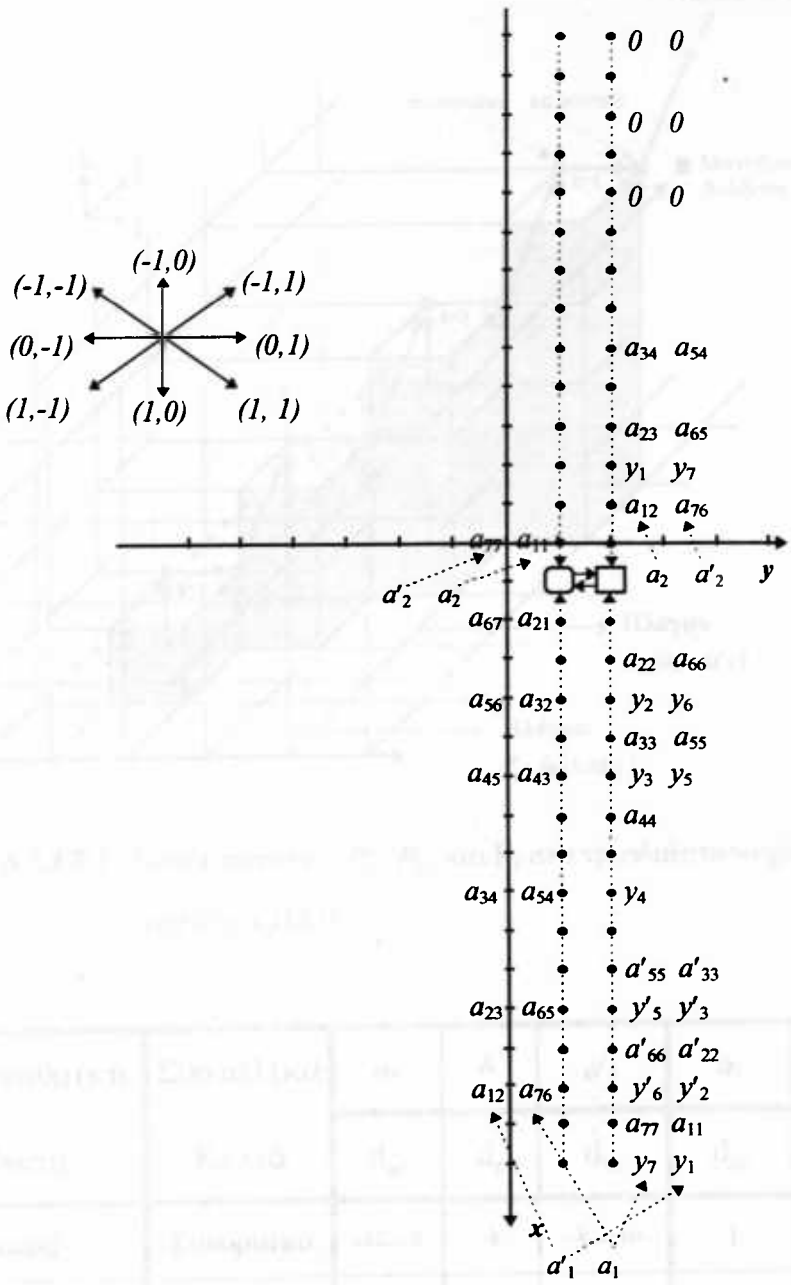
Για την dΦp $\pi_\rho(1,0,1)$, ο συστολικός πίνακας που προκύπτει καθώς και η αντίστοιχη ροή των δεδομένων δίνονται στο σχήμα 6.3.3.2-2.

6.3.3.3 Κατεύθυνση Προβολής $\pi_\rho(1,1,1)$

Για την dΦp $\pi_\rho(1,1,1)$, ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών :

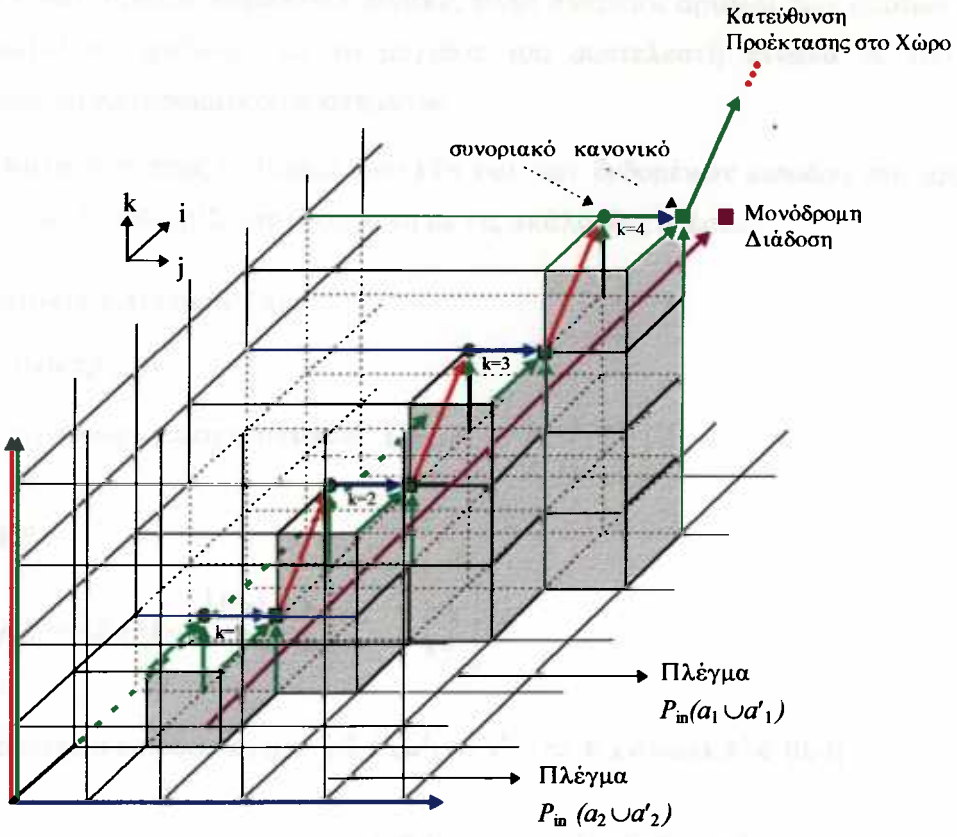
$$u = u(i) = i, \qquad i = 2, \ldots, n+1$$
$$v = v(i,j) = i+j-2, \qquad i = 2, \ldots, n+1, \ j = 1, 2$$
$$w = w(i,k) = i+k-2, \quad i = 2, \ldots, n+1, \ k = 1$$

Γι' αυτή την dΦp η συστολική διάταξη εκτείνεται στο χώρο, όπως φαίνεται στο σχήμα 6.3.3.3-1.



Σχήμα 6.3.3.2-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη $(SA_2, d\Diamond p : \pi_\rho(1,0,1))$

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την $d\Diamond p$ προκύπτουν από τη σχέση (6.3.3.1:1), στην οποία οι τιμές της f προσδιορίζονται με βάση τον πίνακα 6.3.3.3-1.



Σχήμα 6.3.3.3-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στο τρισδιάστατο χώρο προς την $d\Diamond p\ \pi_\rho(1,1,1)$

Αλγοριθμική Φάση	Συστολικά Κελιά	a_1	&	a'_1	a_2	&	a'_2
		d_{ac}	d_c	d_{bc}	d_{ac}	d_c	d_{bc}
Απαλοιφή	Συνοριακό	1,0,...,-k	-k	-k,...,-m	1	-	-
	Κανονικό	1,0,...,-k	-	-k+1,...,-m	1,0,...,-k	-	-k+1,...,-m
Τροποποίηση	Συνοριακό	-	-	-	-	-	-
Σταθερών Όρων	Κανονικό	2,1,...,-k+2	-k+2	-k+2,...,-m+1	2	-	-

Πίνακας 6.10.3-1: Οι Τιμές των Παραμέτρων f_{el} και f_{rhs} για τα Διάφορα Τμήματα Δεδομένων για την $d\Diamond p\ \pi_\rho(1,1,1)$

Τα k και m, στον παραπάνω πίνακα, είναι ακέραιοι αριθμοί των οποίων οι τιμές ποικίλλουν ανάλογα με το μέγεθος του συντελεστή πίνακα A του αρχικά θεωρούμενου γραμμικού συστήματος.

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_3:

- d◊p $\pi_\rho(1,1,1)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix}$

PE:

$$p(i,\ i+j-2,\ i+k-2) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2-k \\ k-j \end{bmatrix}$$

στοιχεία εισόδου a_1 (i, i+j-2, k), a'_1 (i^*, i^*+j^*-2, k^*), όπου $k,k^* \in \{0,-1\}$:

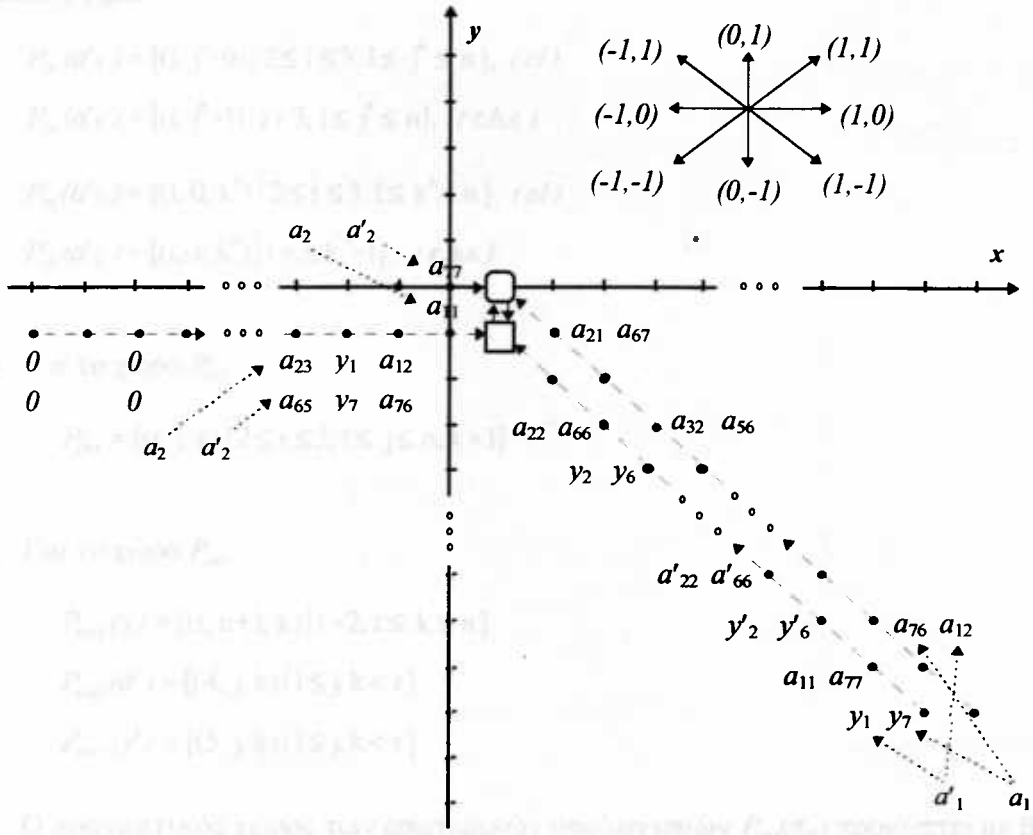
$$\check{p}_{\alpha_1 \& \alpha'_1}\ (i^\#,\ i^\#+j^\#-2,\ 6-2i^\#-j^\#-f_g) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha'_1} = \begin{bmatrix} 3i^\#+j^\#+f_g-6 \\ 8-3i^\#-2j^\#-f_g \end{bmatrix},\ (el \ \&\ r.h.s: d_{ac}, d_c, d_{bc})$$

στοιχεία εισόδου a_2 (i, j, k), a'_2 (i^*, j, k^*), όπου $i,i^* \in \{0,-1\}$:

$$\check{p}_{\alpha_2 \& \alpha'_2}\ (4-j^\#-k^\#-f_g,\ j^\#,\ k^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha'_2} = \begin{bmatrix} 4-j^\#-2k^\#-f_g \\ k^\#-j^\# \end{bmatrix},\ (el \ \&\ r.h.s: d_{ac}, d_c, d_{bc})$$

- ***Προβολικό επίπεδο***

Για την d◊p $\pi_\rho(1,1,1)$, η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.3.3.3-2.



Σχήμα 6.3.3.3-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

(SA_3, d◊p: $\pi_\rho(1,1,1)$)

6.10.4 Κατεύθυνση προβολής $\pi_\rho(0,1,1)$

Για την d◊p $\pi_\rho(0,1,1)$, οι υποχώροι P_{in}, P_{int} και P_{out} πρέπει να αναμορφωθούν, γιατί οι ευθείες οι οποίες ορίζονται από τις αντίστοιχες παραμετρικές εξισώσεις διέρχονται από διαφορετικά σημεία του χώρου. Έτσι,

- Για το χώρο P_{in}:

Πάνω Ρεύμα

$P_{in}(a_1) = \{(i, j, 0) \mid 2 \le i \le 3, 1 \le j \le n\}$, *(el)*

$P_{in}(a_1) = \{(i, j, -1) \mid i = 3, 1 \le j \le n\}$, *(r.h.s.)*

$P_{in}(a_2) = \{(i, 0, k) \mid 2 \le i \le 3, 1 \le k < n\}$, *(el)*

$P_{in}(a_2) = \{(i, -1, k) \mid i = 3, k = 1\}$, *(r.h.s.)*

Κάτω Ρεύμα

$P_{in}(a'_1) = \{(i, j^*, 0) \mid 2 \le i \le 3, 1 \le j^* \le n\}$, *(el)*

$P_{in}(a'_1) = \{(i, j^*, -1) \mid i = 3, 1 \le j^* \le n\}$, *(r.h.s.)*

$P_{in}(a'_2) = \{(i, 0, k^*) \mid 2 \le i \le 3, 1 \le k^* < n\}$, *(el)*

$P_{in}(a'_2) = \{(i, -1, k^*) \mid i = 3, k^* = 1\}$, *(r.h.s.)*

- Για το χώρο P_{int}:

$P_{int} = \{(i, j, k) \mid 2 \le i \le 3, 1 \le j \le n, k = 1\}$

- Για το χώρο P_{out}:

$P_{out}(x) = \{(i, n+1, k) \mid i = 2, r \le k \le n\}$

$P_{out}(a') = \{(4, j, k) \mid 1 \le j, k < r\}$

$P_{out}(y') = \{(5, j, k) \mid 1 \le j, k < r\}$

Ο πραγματικός χώρος των εσωτερικών υπολογισμών $P_{int}(\pi_\rho)$ προκύπτει με βάση τις ακόλουθες παραμετρικές εξισώσεις ευθειών:

$u = u(i) = i, \qquad i = 2,3$

$v = v(j) = j, \qquad j = 1, ..., n$

$w = w(j,k) = j+k-1, \qquad j = 1, ..., n, \; k=1$

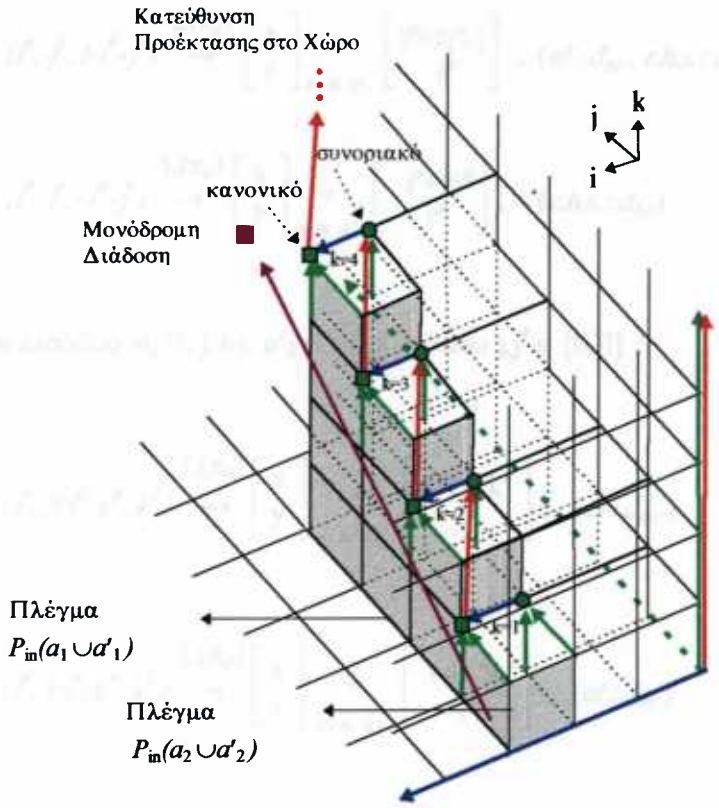
Η συστολική διάταξη εκτείνεται στο χώρο όπως φαίνεται στο σχήμα 6.3.3.4-1

Οι εκτεινόμενες στο χώρο θέσεις των δεδομένων εισόδου γι' αυτή την dôp προκύπτουν από τη σχέση (6.3.2.1:1), ενώ οι παράμετροι f_{el} και f_{rhs} παίρνουν κάθε φορά την κατάλληλη τιμή με βάση τον πίνακα 6.3.3.1-1.

Κατά συνέπεια, οι θέσεις των PEs και των δεδομένων εισόδου στο προβολικό επίπεδο προσδιορίζονται σύμφωνα με τις ακόλουθες σχέσεις:

Συστολική Διάταξη, SA_4:

- dôp $\pi_\rho(0,1,1)$
- πίνακας μετασχηματισμού $L(\pi_\rho) = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}$



Σχήμα 6.3.3.4-1: Προέκταση των P_{in}, P_{int} και Γ_ℓ στον τρισδιάστατο χώρο προς την d◊p $\pi_\rho(0, 1, 1)$

PE :

$$p(i, j, j+k-1) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-k \\ i \end{bmatrix}$$

στοιχεία εισόδου a_1 (i, j, k), a'_1 (i^*, j^*, k^*), όπου $k,k^* \in \{0,-1\}$:

$$\check{p}_{\alpha_1 \,\&\, \alpha'_1}(i^\#, j^\#, 3-i^\#-j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \,\&\, \alpha'_1} = \begin{bmatrix} i^\#+2j^\#-3 \\ i^\# \end{bmatrix}, (el : d_{ac})$$

$$\check{p}_{\alpha_1 \,\&\, \alpha'_1}(i^\#, j^\#, 2-i^\#-j^\#) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \,\&\, \alpha'_1} = \begin{bmatrix} i^\#+2j^\#-2 \\ i^\# \end{bmatrix}, (el : d_c, r.h.s. : d_{ac})$$

$$\check{p}_{\alpha_1 \& \alpha_1'}\ (i^{\#}, j^{\#}, 1\text{-}i^{\#}\text{-}j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha_1'} = \begin{bmatrix} i^{\#}+2j^{\#}-1 \\ i^{\#} \end{bmatrix},\ (el: d_{bc},\ r.h.s.: d_c)$$

$$\check{p}_{\alpha_1 \& \alpha_1'}\ (i^{\#}, j^{\#}, -i^{\#}\text{-}j^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_1 \& \alpha_1'} = \begin{bmatrix} i^{\#}+2j^{\#} \\ i^{\#} \end{bmatrix},\ (r.h.s.: d_{bc})$$

στοιχεία εισόδου a_2 (i, j, k), a'_2 (i^*, j^*, k^*), όπου $j, j^* \in \{0,-1\}$:

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (i^{\#}, 3\text{-}i^{\#}\text{-}k^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 3\text{-}i^{\#}\text{-}2k^{\#} \\ i^{\#} \end{bmatrix},\ (el: d_{ac})$$

$$\check{p}_{\alpha_2 \& \alpha_2'}\ (i^{\#}, 1\text{-}i^{\#}\text{-}k^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 1\text{-}i^{\#}\text{-}2k^{\#} \\ i^{\#} \end{bmatrix},\ (el: d_{bc})$$

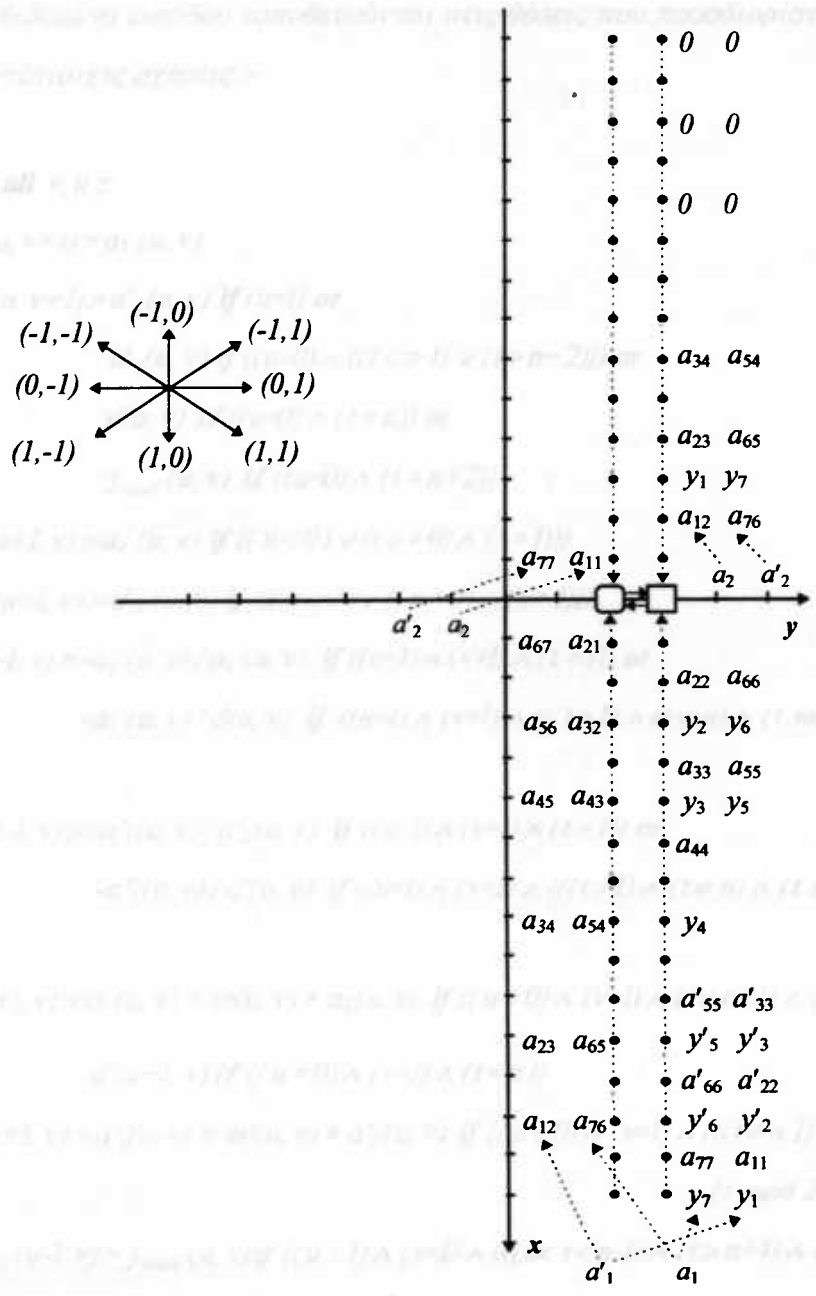
$$\check{p}_{\alpha_2 \& \alpha_2'}\ (i^{\#}, 2\text{-}i^{\#}\text{-}k^{\#}, k^{\#}) \xrightarrow{L(\pi_\rho)} \begin{bmatrix} x \\ y \end{bmatrix}_{\alpha_2 \& \alpha_2'} = \begin{bmatrix} 2\text{-}i^{\#}\text{-}2k^{\#} \\ i^{\#} \end{bmatrix},\ (r.h.s.: d_{ac})$$

- ***Προβολικό επίπεδο***

Για την d◊p $\pi_\rho(0,1,1)$, η συστολική διάταξη που προκύπτει, καθώς και η αντίστοιχη ροή των δεδομένων, δίνονται στο σχήμα 6.3.3.4-2.

6.3.4 Προγραμματιστική Αναπαράσταση στο Επίπεδο Προβολής

Θεωρώντας ότι $u = x$ και $v = y$, η προγραμματιστική αναπαράσταση του αλγόριθμου στο διδιάστατο χώρο, για την d◊p $\pi_\rho(1,1,0)$ (παρόμοια και για όλες τις επιτρεπτές d◊p$_s$) περιγράφεται με τον παρακάτω τρόπο :



Σχήμα 6.3.3.4-2: Ροή Δεδομένων στη Μονοδιάστατη Συστολική Διάταξη

$(SA_4, d\Diamond p : \pi_p(0,1,1))$

Program AEB&F

initially : *t:=0*

< Τα δεδομένα εισόδου τοποθετούνται στις θέσεις που προσδιορίστηκαν από τις αντίστοιχες σχέσεις >

assign :

*< **for all** v, u ::*

|| $a_1(u, v+1) := a_1(u, v)$

|| $a'_1(u, v+1) := a'_1(u, v)$ **if** $(u=1)$ **or**

$a'_1(u, v)$ **if** $((u=0) \wedge ((t<n-1) \vee (t>n+2)))$ **or**

$d(u, v)$ **if** $((u=0) \wedge (t=n))$ **or**

$y_{mod}(u, v)$ **if** $((u=0) \wedge (t=n+2))$

|| $a_2(u+1, v) := a_2(u, v)$ **if** $((u<0) \vee ((u=0) \wedge (t=1)))$

|| $a'_2(u+1, v) := a'_2(u, v)$ **if** $((u<0) \vee ((u=0) \wedge (t=1)))$

|| $m(u-1, v) := -a_1(u, v) / a_2(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t=1))$ **or**

$-a_1(u, v) / d(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (((t>1) \wedge (t \neq n) \wedge (t \bmod 2 = 1)) \vee (t=n+1)))$

|| $m'(u-1, v) := -a'_1(u, v) / a'_2(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t=1))$ **or**

$-a'_1(u, v) / d'(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (((t>1) \wedge (t \neq n) \wedge (t \bmod 2 = 1)) \vee (t=n+1)))$

|| $d(u+1, v) := a_1(u, v) + m(u, v) * a_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t \neq n+1) \wedge (t \bmod 2 = 0))$ **or**

$d'(u+1, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=n))$

|| $d'(u+1, v) := a'_1(u, v) + m'(u, v) * a'_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (((t \neq n-1) \wedge (t \neq n+1) \wedge (t \bmod 2 = 0)) \vee (t=n)))$

|| $y_{mod}(u-1, v) := y_{mod}(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (((2<t<n-1) \wedge (t>n+1) \wedge (t \bmod 2 = 0)) \vee (t=n)))$

|| $y'_{mod}(u-1, v) := y'_{mod}(u, v)$ **if** $((u=1) \wedge (v=1) \wedge ((((2<t<n-1) \vee (t>n+1)) \wedge (t \bmod 2 = 0)) \vee (t=n)))$

|| $y_{mod}(u+1, v) := a_1(u, v) + m(u, v) * a_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=3))$ **or**

$a_1(u, v) + m(u, v) * y_{mod}(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (((3<t<n) \vee$

$(t>n+2)) \wedge (t \bmod 2 = 1)) \vee (t=n+1))$ **or**

$y'_{mod}(u+1, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=n+2))$

|| $y'_{mod}(u+1, v) := a'_1(u, v) + m'(u, v) * a'_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=3))$ **or**

$a'_1(u, v) + m'(u, v) * y'_{mod}(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t>3) \wedge (t \neq n)$

$\wedge (t \bmod 2 = 1))$

|| $x(u+1, v) := y_{mod}(u, v) / d(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t>n+1) \wedge (t \bmod 2 = 0))$

|| $x'(u+1, v) := y'_{mod}(u, v) / d'(u, v)$ **if** $((u=1) \wedge (v=1) \wedge (t>n+3) \wedge (t \bmod 2 = 0))$

|| $d(u, v+1) := a_1(u, v) + m(u, v) * a_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t \neq n-1) \wedge (t<n) \wedge$

$(t \bmod 2 = 0))$

|| $d'(u, v+1) := a'_1(u, v) + m'(u, v) * a'_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (((t \neq n-1) \wedge (t<n) \wedge$

$(t \bmod 2 = 0)) \vee (t=n)))$

|| $y_{mod}(u, v+1) := a_1(u, v) + m(u, v) * a_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=3))$ **or**

$a_1(u, v) + m(u, v) * y_{mod}(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (3<t<n) \wedge$

$(t \bmod 2 = 1))$

|| $y'_{mod}(u, v+1) := a'_1(u, v) + m'(u, v) * a'_2(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (t=3))$ **or**

$a'_1(u, v) + m'(u, v) * y'_{mod}(u, v)$ **if** $((u=0) \wedge (v=1) \wedge (((3<t<n) \wedge$

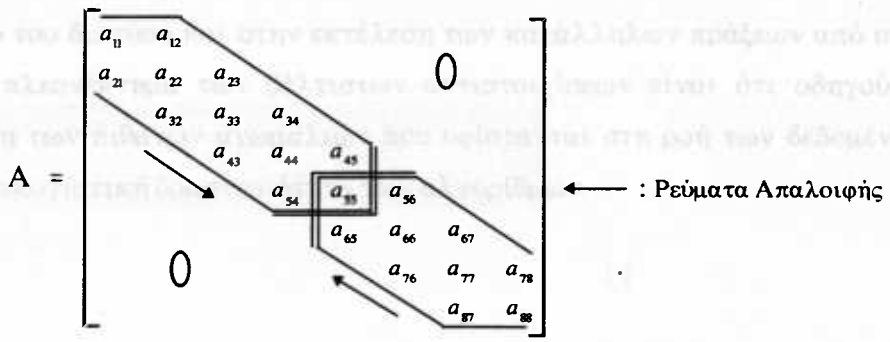
$(t \bmod 2 = 1)) \vee (t=n+2)))$ >

// $t := t+1$

end. {AEB&F}

6.4 Συμπεράσματα

Οι αλγοριθμικές διαδικασίες AEMDR και AEB&F, παρουσιάστηκαν στις Παραγράφους 6.2 και 6.3, αντίστοιχα, θεωρώντας ότι το μέγεθος του συντελεστή πίνακα Α είναι περιττός αριθμός. Στο βαθμό που το μέγεθος του συντελεστή πίνακα θεωρηθεί ότι είναι άρτιος αριθμός, τα ρεύματα των δεδομένων διαμορφώνονται όπως φαίνεται στο σχήμα 6.4-1.



Σχήμα 6.4-1: Μαθηματική Μορφοποίηση του Πάνω και Κάτω Ρεύματος Δεδομένων (για n=8)

Ο συγκεκριμένος τρόπος διαμόρφωσης των ρευμάτων δεδομένων, είτε για περιττό ή για άρτιο n, επιτρέπει να διατηρείται μία μόνο γραμμή και όχι ένας (2x2) πίνακας στο κέντρο. Το γεγονός αυτό απλοποιεί τη διαδικασία υλοποίησης των αλγορίθμων και προυποθέτει ένα μόνον επιπλέον χρονικό βήμα για την επεξεργασία του κεντρικού στοιχείου και από το κάτω ρεύμα δεδομένων.

Οι δύο παραπάνω αλγοριθμικές διαδικασίες, χρησιμοποιήθηκαν για την παρουσίαση του εργαλείου SDT. Το εργαλείο αυτό εστιάζει στη δυνατότητα αναπαράστασης συστολικών αλγορίθμων κάνοντας χρήση μιας διανυσματικής προσέγγισης η οποία εκφράζει τις εξαρτήσεις μεταξύ των δεδομένων κατά τη διάρκεια της εξέλιξης της αλγοριθμικής διαδικασίας. Το γεγονός αυτό επιτρέπει το 'ξεδίπλωμα' του αλγόριθμου στον τρισδιάστατο χώρο προς κάποια *επιτρεπτή* κατεύθυνση προβολής με σκοπό την απαλοιφή των πιθανών ανωμαλιών στη ροή των δεδομένων και στην υπολογιστική δραστηριότητα. Κατ' επέκταση επιτυγχάνεται μία περαιτέρω ανάλυση του αλγόριθμου, η οποία έχει μαθηματική βάση και αποσκοπεί, κυρίως, στην εύρεση ενός κατάλληλου γραφήματος που θα απεικονίζει τη διάρθρωση των επεξεργαστών η οποία θα επιλύει τον αλγόριθμο.

Γενικά, η δυνατότητα άμεσης έκφρασης των αλγορίθμων υπό μορφή γραφημάτων, τα οποία προσδιορίζουν την πλέον κατάλληλη δομή φυσικών επεξεργαστών για την επίλυση κάποιου προβλήματος, αποτελεί ένα από τα πλέον δύσκολα και ενδιαφέροντα προβλήματα τα οποία απασχολούν τις τελευταίες δεκαετίες την επιστημονική κοινότητα. Στο βαθμό που επιτυγχάνεται μία τέτοια κατάλληλη αντιστοίχιση μεταξύ αλγορίθμων και διατάξεων επεξεργαστών, η εκτέλεση οποιουδήποτε αλγόριθμου συνίσταται απλά στη ροή των δεδομένων

διαμέσω του δικτύου και στην εκτέλεση των κατάλληλων πράξεων από αυτό. Το βασικό πλεονέκτημα των βέλτιστων αντιστοιχίσεων είναι ότι οδηγούν στην εξάλειψη των πιθανών ανωμαλιών που υφίστανται στη ροή των δεδομένων και στην υπολογιστική δραστηριότητα των αλγορίθμων.

Παράρτημα Κ:4 - §4.5

Παράλληλη Προσομοίωση της Συστολικής Μηχανής Πολλαπλών Λειτουργιών (MFE)

Η παράλληλη προσομοίωση της συστολικής μηχανής που επιτυγχάνει την ελαχιστοποίηση του εύρους ζώνης και της κατατομής ενός αραιού πίνακα, έχει υλοποιηθεί κάνοντας χρήση του δυναμικού περιβάλλοντος της Multi-Pascal. Ο βασικός λόγος που οδήγησε στη χρήση του προσομοιωμένου αυτού παράλληλου εργαλείου υπήρξε η παρεχόμενη δυνατότητα χρήσης ενός σχετικά μεγάλου πλήθους *'επεξεργαστών'* για την προσομοίωση της συστολικής μηχανής, η οποία εκ των πραγμάτων προυποθέτει τη χρήση μιας αρχιτεκτονικής πολυεπεξεργασίας. Με δεδομένο το γεγονός ότι το μόνο τέτοιο παράλληλο σύστημα που υπάρχει στη διάθεσή μας είναι το Silicon Graphics (Εθνικό Μετσόβειο Πολυτεχνείο), το οποίο διαθέτει 16 μόνον επεξεργαστές, προέκυψε η ανάγκη της προσομοίωσης προκειμένου για την επεξεργασία μεγάλου μεγέθους αραιών πινάκων. Από την άλλη μεριά, τα παράλληλα συστήματα Parsytec GC_{el} (Εργαστήριο Υπολογιστών Υψηλών Επιδόσεων Αθηνών) και Paragon (Εθνικό Μετσόβειο Πολυτεχνείο), τα οποία διαθέτουν 512 και 48 επεξεργαστές, αντίστοιχα, παρόλο που εξαλείφουν, σχετικά, το πρόβλημα του περιορισμού όσον αφορά στο διαθέσιμο πλήθος επεξεργαστών, είναι συστήματα κατανεμημένης μνήμης. Όμως, όπως ήδη αναφέρθηκε, οποιαδήποτε συστολική μηχανή, απαιτεί μία μορφή κεντρικού

ελέγχου και για το λόγο αυτό, από προγραμματιστικής άποψης, βρίσκεται πιο κοντά στη φιλοσοφία ενός συστήματος πολυεπεξεργασίας.

Η Multi-Pascal λόγω του ότι προσφέρει ένα προσομοιωμένο παράλληλο περιβάλλον, επιτρέπει την επιλογή οποιασδήποτε από τις δύο προαναφερθείσες παράλληλες αρχιτεκτονικές με δυνατότητα προσομοίωσης ενός πλήθους επεξεργαστών που κυμαίνονται από 1 έως 256. Το δυναμικό αυτό περιβάλλον παρέχει μηχανισμούς παρακολούθησης της απόδοσης των παράλληλων προγραμμάτων που αναπτύσσονται. Επιπλέον, υφίστανται χρονικοί περιορισμοί οι οποίοι αφορούν, κυρίως, στο χρόνο δημιουργίας των διεργασιών, στον ανταγωνισμό των επεξεργαστών για την κατοχή της καταμεριζόμενης μνήμης και στις καθυστερήσεις για την μεταξύ των επεξεργαστών επικοινωνία. Ένα μεγάλο πλήθος πειραματικών αποτελεσμάτων αποδεικνύουν ότι η προσομοίωση είναι αρκετά ρεαλιστική σε σχέση με τα αποτελέσματα που επιτυγχάνονται σε πραγματικά παράλληλα συστήματα.

Στη συνέχεια, παρατίθεται ο κώδικας, σε Multi-Pascal, που αντιστοιχεί στην υλοποίηση μίας συστολικής μηχανής πολλαπλών λειτουργιών για την ελαχιστοποίηση του εύρους ζώνης και της κατατομής ενός αραιού συμμετρικού πίνακα.

```
program systolic_bpr;

architecture shared(16);

const n=16;

type
  pin=array[1..n,1..n] of integer;
  pin1=array[1..n] of integer;
  chan=array[1..2] of channel of integer;
  chan1=array[1..n] of channel of integer;
  chan2=array[1..2*n] of channel of integer;


var
  lev,reg,botchan,verts,singlecell:chan1;  topchan:chan2;
  a,sp,sp1,levels,test:pin; fnew,f,degr,temp,nums,nums1,fnums,vertices:pin1;
  bw,pf,bw1,pf1,in,pl,pl1,i,j,k,v,counter,gl,vertex,lastlev:integer;
  c:spinlock;  first,f1,f2,f3:boolean;
```

```
procedure pipenet(var f:pin1; var bw,pf:integer);
var   ch:chan;
        j:integer;
(* Computes through a pipenet the bandwidth and the profile of a given matrix.
   The two parameters are computed simultaneously *)

procedure sub(f:pin1);
var i:integer;

begin
 for i:=1 to n do
 begin
   ch[1]:=i-f[i];
   ch[2]:=i-f[i]
 end;
end;

procedure max(var bw:integer);
var i,rw:integer;
begin
  bw:=0;
  for i:=1 to n do
  begin
    rw:=ch[1];
    if rw>bw then
      bw:=rw;
  end;
end;

procedure add(var pf:integer);
var i,rw:integer;
begin
 pf:=0;
 for i:=1 to n do
 begin
   rw:=ch[2];
   pf:=pf+rw;
 end;
end;

begin (* pipenet *)
 fork sub(f);
 fork max(bw);
 fork add(pf);
end;
```

```
procedure comp_cell(i,j,v:integer);
var a,b:integer;

(* Simulates the basic cell operation of the reconfigurable systolic array which computes
   a level structure *)

begin
 if (v<>-1) then  (* Normal Cell *)
 begin
   a:=topchan[j];
   b:=botchan[v];
   while (a<>-1) and (b<>-1) do
   begin
     if (a>b) then
     begin
       topchan[j+1]:=b;
       reg[i+1]:=b;
       lev[i+1]:=b;
       if (i>=2) then verts[i+1]:=v;
       lock(c);
         counter:=counter+1;
       unlock(c);
       b:=botchan[v]
     end
     else
     begin
       if (a=b) then b:=botchan[v];
       topchan[j+1]:=a;
       a:=topchan[j]
     end;
   end;
   while (a<>-1) do
   begin
     topchan[j+1]:=a;
     a:=topchan[j];
   end;
   while (b<>-1) do
   begin
     topchan[j+1]:=b;
     reg[i+1]:=b;
     lev[i+1]:=b;
     if (i>=2) then verts[i+1]:=v;
     lock(c);
       counter:=counter+1;
     unlock(c);
     b:=botchan[v];
   end;
   topchan[j+1]:=-1
 end
```

```
 else       (* Boundary Cell *)
  begin
   if (i=1) then
   begin
    a:=topchan[j];
    while (a<>-1) do
    begin
     topchan[j+1]:=a;
     a:=topchan[j];
    end;
   end
   else
    begin
     a:=topchan[j];
     b:=lev[i-1];
     while (a<>-1) and (b<>-1) do
      if (a<=b) then
       if (a<b) then
       begin
        topchan[j+1]:=a;
        a:=topchan[j]
       end
       else
       begin
        a:=topchan[j];
        b:=lev[i-1];
       end;
      while (a<>-1) do
       begin
        topchan[j+1]:=a;
        a:=topchan[j];
       end;
    end;
    topchan[j+1]:=-1;
    reg[i+1]:=-1;
    lev[i+1]:=-1;
    verts[i+1]:=-1;
  end;
end; (* comp_cell *)


procedure comparison_cell(i,j,r,n:integer; var first,f1,f2,f3:boolean;  var ch:chan2);
var a,b,t,t1:integer;

(* Simulates the basic cell operation of the systolic array which executes the taxonomy
   procedure *)
```

```
begin
 if ((i<>1) and ((i mod 2)<>0)) then
 begin
  if (j mod (n-1)) = 0 then
   ch[n+1]:=ch[n];
  if (j mod n) = 1 then
   ch[j]:=ch[2*n]
 end;
 if (ch[j]?) and (ch[2*n-j]?) then
 begin
  a:=ch[j];
  b:=ch[2*n-j];
  ch[j]:=a;
  ch[2*n-j]:=b;
  if a>b then
  begin
   ch[j+1]:=ch[j];
   ch[2*n-j+1]:=ch[2*n-j];
   if f1 then
   begin
    if f3 then
    begin
     t1:=vertices[j];
     vertices[j]:=vertices[j+1];
     vertices[j+1]:=t1;
    end;
    t:=temp[j];
    temp[j]:=temp[j+1];
    temp[j+1]:=t
   end
   else if f2 then
   begin
    t:=test[r,j];
    test[r,j]:=test[r,j+1];
    test[r,j+1]:=t
   end
  end
  else begin
   ch[j+1]:=ch[2*n-j];
   ch[2*n-j+1]:=ch[j]
  end;
 end;
 if first then
  if (j mod (n-1)) = 0 then
  begin
   ch[n+1]:=ch[n];
   first:=false;
  end;
end; (* comparison_cell *)
```

```
procedure vertex_sort(i:integer);
var   k,val,n,j,m:integer;
      ch:chan2;

(* Channel variables are used as delay elements. These variables have
   to be initialized before the systolic sorting procedure starts
   Moves the channel contents into the appropriate delay elements.
   Sort the level elements and locate them into the appropriate row (level)
   of the table containing the level structure. *)

begin
 val:=lev[i]; k:=1;
 while (val<>-1) do
 begin
   if (i=2) then levels[i,k]:=val
   else ch[2*k-1]:=val;
   k:=k+1;
   val:=lev[i];
 end;

 n:=k-1;

 if (i=2) then
   for j:=1 to n do
     if (j mod 2) <> 0 then
       ch[j]:=degr[temp[j]]
     else  ch[2*n-j+1]:=degr[temp[j]];

 first:=false;
 if (n mod 2) = 1 then
 begin
   first:=true;
   m:=n
 end
 else m:=n-1;

 f2:=false; f3:=false;

 for k:=1 to m do
   forall j:=1 to n-1 do
     if (i=2) then
     begin
       f1:=true;
       comparison_cell(k,j,0,n,first,f1,f2,f3,ch)
     end
     else begin
         f1:=false;
         comparison_cell(k,j,0,n,first,f1,f2,f3,ch);
       end;
```

```
 if (i=2) then
 begin
  for j:=1 to n do
  begin
    gl:=gl+1;
    nums[gl]:=temp[j];
    nums1[temp[j]]:=gl;
    temp[j]:=0;
  end;
  for k:=1 to n do
    if (ch[k]?) then val:=ch[k]
    else val:=ch[2*n-k+1];
 end
 else begin
      for j:=1 to n do
        if (ch[j]?) then
          levels[i,j]:=ch[j]
        else levels[i,j]:=ch[2*n-j+1];
     end;

(* Relocates the sorted elements into the original channel from which
  they are taken *)

  j:=1;
  while (levels[i,j]<>0) do
  begin
    lev[i]:=levels[i,j];
    j:=j+1;
  end;
  lev[i]:=-1;
end; (* vertex_sort *)


procedure level(var i,k:integer);
var j,val:integer;

(* Stores the contents of the first level into table levels or
  stores the contents of every other level into a temporary
  linear table temp *)

begin
  val:=lev[i]; k:=1;
  while (val<>-1) do
  begin
    if (i=1) then levels[i,k]:=val
    else temp[k]:=val;
    k:=k+1;
    val:=lev[i];
  end;
  k:=1;
```

```
  if (i=1) then
    while (levels[i,k]<>0) do
    begin
      lev[i]:=levels[i,k];
      k:=k+1
    end
  else
    while (temp[k]<>0) do
    begin
      lev[i]:=temp[k];
      k:=k+1;
    end;
  lev[i]:=-1;
end; (* level *)


procedure taxonomy (i,n:integer);
var val,same,k,it,m,j,l:integer;
     max:pin1;  nch:chan2;

(* Performs the systolic taxonomy procedure of the vertices of each level
  in increasing degree order *)

begin
 for j:=1 to n do
 begin
   val:=verts[i];
   vertices[j]:=val;
   if (j mod 2) <> 0 then
     nch[j]:=nums1[val]
   else  nch[2*n-j+1]:=nums1[val];
 end;
 val:=verts[i];

 first:=false;
 if (n mod 2) = 1 then
 begin
   first:=true;
   m:=n
 end
 else m:=n-1;

 f1:=true; f2:=false; f3:=true;

 for k:=1 to m do
   forall j:=1 to n-1 do
      comparison_cell(k,j,0,n,first,f1,f2,f3,nch);
```

```
for k:=1 to n do
begin
  if (nch[k]?) then val:=nch[k]
  else val:=nch[2*n-k+1];
  verts[i]:=vertices[k];
  vertices[k]:=0;
end;
verts[i]:=-1;

(* Separates the vertices that are produced from different cells in order
  to sort them in increasing degree order individually *)

same:=verts[i]; val:=same;
k:=1; l:=1; j:=1;
while (val<>-1) do
begin
  if (same<>val) then
  begin
    same:=val;
    max[k]:=l-1;
    if (l-1>1) then singlecell[k]:=-1;
    k:=k+1;
    l:=1;
  end;
  singlecell[k]:=degr[temp[j]];
  test[k,l]:=temp[j];
  temp[j]:=0;
  l:=l+1;
  j:=j+1;
  val:=verts[i];
end;
max[k]:=l-1;
if (l-1>1) then singlecell[k]:=-1;

for j:=1 to k do
  if (max[j]>=2) then
  begin
    for l:=1 to max[j] do
      if (l mod 2) <> 0 then
        nch[l]:=singlecell[j]
      else  nch[2*max[j]-l+1]:=singlecell[j];

    val:=singlecell[j];

    first:=false; f1:=false; f2:=true; f3:=false;
    if (max[j] mod 2) = 1 then
    begin
      first:=true;
      m:=max[j]
    end
    else m:=max[j]-1;
```

```
  for l:=1 to m do
    forall it:=1 to max[j]-1 do
      comparison_cell(l,it,j,max[j],first,f1,f2,f3,nch);

   for l:=1 to max[j] do
    if (nch[l]?) then val:=nch[l]
    else val:=nch[2*max[j]-l+1]
 end
 else  val:=singlecell[j];

(* Gives each vertex of the specific level a different number *)

 for j:=1 to k do
 begin
  l:=1;
  while (test[j,l]<>0) do
  begin
   gl:=gl+1;
   nums[gl]:=test[j,l];
   nums1[test[j,l]]:=gl;
   test[j,l]:=0;
   l:=l+1;
  end;
 end;
end; (* taxonomy *)


procedure new_matrix(var sp,sp1:pin; var fnew:pin1);
var i,j:integer;

(* Formulates the compacted matrix according to the new numbering *)

begin
 for i:=1 to n do
  for j:=1 to i do
    if sp[i,j]<>0 then
    begin
      sp1[nums1[i],nums1[j]]:=1;
      if i<>j then sp1[nums1[j],nums1[i]]:=1;
    end;
 first:=true;
 for i:=1 to n do
 begin
  first:=true;
  for j:=1 to i do
    if (sp1[i,j]<>0) and (first) then
    begin
     fnew[i]:=j; first:=false;
    end;
 end;
end; (* new_matrix *)
```

```
procedure min(var bw,bw1,pf,pf1:integer; var fnums,nums1:pin1);

(* Selects the the numbering that produces the minimum possible bandwidth and profile *)

begin
 if ((bw1<bw) or ((bw1<=bw) and (pf1<pf))) then
 begin
  bw:=bw1; pf:=pf1;
  fnums:=nums1;
 end;
end; (* min *)


procedure init (i:integer; var sp1:pin);
var k,j,val:integer;

(* Initializes the table sp1 containing the resulted compacted matrix
  for a given level structure *)

begin
 for k:=1 to n do
  for j:=1 to k do
   if sp1[k,j]<>0 then
   begin
    sp1[k,j]:=0;
    if k<>j then sp1[j,k]:=0;
   end;

(* Initializes the channel variables topchan[j] containing the top stream,
  i.e. all the vertices that have been assigned into a level *)

 for j:=1 to 2*n do
  if (topchan[j]?) then
  begin
   val:=topchan[j];
   while (val<>-1) do
    val:=topchan[j];
  end;

(* Initializes the channel variables botchan[v] containing the adjecant
  vertices of each vertex v *)

 for j:=1 to n do
  if (botchan[j]?) then
  begin
   val:=botchan[j];
   while (val<>-1) do
    val:=botchan[j];
  end;
```

```
 for j:=1 to n do
 begin
  k:=1;
  while (a[j,k]<>0) do
  begin
   botchan[j]:=a[j,k];
   k:=k+1;
  end;
  botchan[j]:=-1;
 end;

(* Initializes the table 'levels' containing a specific level structure *)

 for j:=1 to i do
 begin
  k:=1;
  while (levels[j,k]<>0) do
  begin
   levels[j,k]:=0;
   k:=k+1;
  end;
 end;

(* Initializes the channel variables 'lev[i]' and 'reg[i]' containing the levels
  of a specific level structure *)

 for j:=1 to i do
 begin
  if (lev[j]?) then
  begin
   val:=lev[j];
   while (val<>-1) do
    val:=lev[j];
  end;
  if (reg[j]?) then
  begin
   val:=reg[j];
   while (val<>-1) do
    val:=reg[j];
  end;
 end;
end; (* init *)
```

```
begin  (* main *)

(* Reads the input data and construct the connectivity matrix.
   Each row of the connectivity matrix is stored in a channel type variable *)

 for i:=1 to n do
 begin
  first:=true; pl:=0; k:=1;
  for j:=1 to n do
  begin
   read(sp[i,j]);
   if (sp[i,j]<>0) and (i<>j) then
   begin
    botchan[i]:=j;
    pl:=pl+1;
    a[i,k]:=j;
    k:=k+1;
   end;
   if (sp[i,j]<>0) and (first) then
   begin
    f[i]:=j; first:=false;
   end;
  end;
  botchan[i]:=-1;
  degr[i]:=pl;
  readln;
 end;

(* Computes the bandwidth and profile *)

  fork pipenet(f,bw,pf);

(* Initializes the channel variables *)

 for vertex:=1 to n do
 begin
  i:=1; j:=1; counter:=1;
  reg[i]:=vertex; reg[i]:=-1;
  lev[i]:=vertex; lev[i]:=-1;
  topchan[j]:=vertex; topchan[j]:=-1;
  nums[1]:=vertex; gl:=1; nums1[vertex]:=1; pl1:=0;

(* Computes the level structure and the numbering *)

  v:=reg[i];
  while (v>=-1) do
  begin
   fork comp_cell(i,j,v);
   j:=j+1;
```

```
    if (v=-1) then
    begin
     level(i,k);
     if (i>=2) then vertex_sort(i);
     if (i>2) then taxonomy(i,k-1);
     i:=i+1;
    end;
    if (counter<n) then v:=reg[i]
    else v:=-2;
  end;

  level(i,k);
  taxonomy(i,k-1);

(* Reverses the resulted numbering *)

  for i:=1 to n do nums1[i]:=n-nums1[i]+1;

(* Computes the compacted matrix *)

  new_matrix(sp,sp1,fnew);

  fork pipenet(fnew,bw1,pf1);

  init(i,sp1);

  min(bw,bw1,pf,pf1,fnums,nums1);

end; (* for vertex:=1 to n do *)

end.   (* systolic_bpr *)
```

Βιβλιογραφία & Παραπομπές

[1] Akhras, G. & Dhatt, G. (1976). *An Automatic Node Relabelling Scheme for Minimizing a Matrix or Network Bandwidth*, International Journal for Numerical Methods in Engineering, v. 10, pp. 787-797.

[2] Barada, H. & El-Amawy, (1993). *A Methodology for Algorithm Regularization and Mapping into Time Optimal VLSI Arrays*, Parallel Comput. (19), pp. 33-61.

[3] Barnard, S.T., Pothen, A. & Simon, H. (1995). *A Spectral Algorithm for Envelope Reduction of Sparse Matrices*, Numerical Linear Algebra with Applications, v. 2(4) pp. 317-334.

[4] Bartzi, A.A. & Bekakos, M.P. (1997). *A Symbolic Descriptive Tool (SDT) Supporting Systolic Program Representations*, AUEB INFO TR : 11, August.

[5] Bartzi, A.A. & Bekakos, M.P. (1997). *A Symbolic Systematic Methodology for Optimal Systolic Architectures and Computations*, AUEB INFO TR : 10, August.

[6] Bartzi, A.A. & Bekakos, M.P. (1997). *A Unidirectional Spatial Representation of the Area Efficient Main Diagonal Redirection Algorithm*, AUEB INFO TR : 12, August.

[7] Bartzi, A.A., Bekakos, M.P. & Lipitakis, E.A. (1996). *Compaction of Sparse Matrices Using a Multilevel Pipelined Approach*, AUEB INFO TR : 2, July.

[8] Bartzi, A.A., Bekakos, M.P. & Lipitakis, E.A. (1996). *Compaction of Sparse Matrices Using a Multilevel Pipelined Approach*, Proc. of the 3rd Hellenic European Conference on Mathematics and Informatics, Hellenic European Research on Mathematics and Informatics, p.p. 259-272.

[9] Bekakos, M.P. & Bartzi, A.A. (1996). *A New Algorithm for the Systolic Reduction of Sparse Matrices on a 2D-Mesh*, Proc. of the 3rd Hellenic European Conference on Mathematics and Informatics, Hellenic European Research on Mathematics and Informatics, p.p. 273-281.

[10] Bekakos, M.P. & Bartzi, A.A. (1996). *A New Algorithm for the Systolic Reduction of Sparse Matrices on a 2D-Mesh*, AUEB INFO TR : 4, July.

[11] Bekakos, M.P. & Bartzi, A.A. (1996). *Systolic Bandwidth and Profile Reduction of Sparse Matrices on Pipenets*, AUEB INFO TR : 6, August.

[12] Bekakos, M.P. & Bartzi, A.A. (1996). *Systolic Bandwidth and Profile Reduction of Sparse Matrices on Pipenets*, 2nd World Congress of Nonlinear Analysts, Athens, Greece.

[13] Bekakos, M.P. & Evans, D.J. (1986). *Single Stage Computational 'Dewavefronts' for the Implementation of the QIF Algorithm*, TR : 282, May, L.U.T., U.K.

[14] Bekakos, M.P. & Evans, D.J. (1986). *Systolic LU-factorization 'Dequeues' for Tridiagonal Systems*, Intern. J. Computer Math., v. 25, pp 299-320, L.U.T., U.K.

[15] Bekakos, M.P. & Evans, D.J. (1986). *The Exposure and Exploitation of Parallelism on Fifth Generation Computer Systems*, Proc. of the Int. Conf. on Parallel Computing '85, (eds.) M. Feilmeir, G. Joubert and U. Schendel, pp. 425-442, Elsevier Science Pub., B.V., North-Holland.

[16] Bekakos, M.P. & Evans, D.J. (1987). *A Rotating and Folding Algorithm Using a Two-Dimensional 'Systolic' Communication Geometry*, Parallel Computing, (4), pp. 221-228, Elsevier Science Pub., B.V., North-Holland.

[17] Bekakos, M.P. & Kouniakis, C.T. (1996). *A New Main Diagonal Redirection Systolic Tridiagonal Linear System Solver*, Kuwait Journal of Science and Engineering, v. 23, pp. 185-199, Kuwait.

[18] Bekakos, M.P. (1993). *A Program Representation for the R&F LU-Decomposition Systolic Algorithm*, Proc. of the 3rd Int. Conf. on Applications of Supercomputers in Engineering, pp. 367-376, WIT, Bath, U.K.

[19] Bekakos, M.P. (1995). *A Notational Approach to Formulation of Systolic Array Programs*, Parallel Computing 21, Elsevier Science Pub. B.V., North-Holland, pp. 619-626.

[20] Bekakos, M.P., Lipitakis, E.A. & Efremides, O.B. (1994). *Parallel Exploitation of Multiple Pipes Arrangements on Mesh Architectures*, 2nd Hellenic European Conf. on Mathematics and Informatics, Hellenic European Research on

Mathematics and Informatics, pp. 855-868, Athens, Greece.

[21] Biggs, N. (1993). *Algebraic Graph Theory*, Cambridge University Press (2nd Edition).

[22] Brawer, S. (1989). *Introduction to Parallel Programming*, Academic Press, U.S.A.

[23] Capello, R.P. & Stiegliz, K. (1984). *Selecting Systolic Designs Using Linear Transformations of Space-Time*, Proc. of the SPIE Symp. 549, Real Time Signal Processing v. 2, pp. 75-85.

[24] Chandy, K. & Misra, J. (1986). *Systolic Algorithms as Programs*, Distributed Computing, pp. 177-183.

[25] Chartrand, G. & Oellermann, O.R. (1993). *Applied and Algorithmic Graph Theory*, Singapore : McGraw-Hill.

[26] Chen, M.C. (1985). *Synthesizing Systolic Designs*, Proc. of the 2nd Int. Symb. VLSI Technol. Syst. Appl., Taipei, Taiwan.

[27] Cheng, K.Y. (1973). *Minimizing the Bandwidth of Sparse Symmetric Matrices*, Computing 11, pp. 27-30, 103-110.

[28] Chinn, P. Z., Chvatalova, J., Dewdney, A.K. & Gibbs, N.E. (1982). *The Bandwidth Problem for Graphs and Matrices - A survey*, J. of Graphs Theory, 6 pp. 223-254.

[29] Codenotti, B. & Leoncini, M. (1993). *Introduction to Parallel Processing*, New York: Addison-Wesley.

[30] Coffin, M.H. (1992). *Parallel Programming : A New Approach*, Prentice Hall.

[31] Cuthill, E. & McKee, J. (1969). *Reducing the Bandwidth of a Sparse Symmetric Matrix*, Proc. of 24th Nat. Conf. ACM, pp. 157-172.

[32] Delosme, J.M. & Ipsen, I.C.F. (1985). *An Illustration of a Methodology for the Construction of Efficient Systolic Architectures in VLSI*, Proc. of the 2nd Int. Symb. VLSI Technol. Syst. Appl., Taipei, Taiwan, pp. 268-273.

[33] Efremides, O.B. & Bekakos, M.P. (1995). *A Parallel Approach to the Bandwidth Minimization (BM) problem*, 3rd Balkan Conference on Operational Research, Thessaloniki, Greece.

[34] Esonu, M.O., Al-Khalili, J., Hariri, S. & Al-Khalili, D. (1992). *Systolic Arrays : How to Choose Them*, IEE Proc 139 (3), pp. 179-188.

[35] Evans, D.J. & Megson, G.M. (1986). *The Triangularization of a Symmetric Tridiagonal Matrix*, Internal Report 324, L.U.T., U.K., October.

[36] Feldman, J.A. (1979). *High Level Programming for Distributed Computing*, Communications of the ACM 22, 6 (June), pp. 341-346.

[37] Foulds, L.R. (1994). *Graph Theory Applications*, New York: Springer-Verlag.

[38] Garey, M.R., Graham, R.L., Johnson, D.S. & Knuth, D.E. (1978). *Complexity Results for Bandwidth Minimization,* SIAM J. Appl. Math. 34 pp. 477-495.

[39] Gibbs, N.E., Poole, W.G. & Stockmeyer, P.K. (1976). *An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix*, SIAM J. on Numer. Anal., v. 13 pp. 236-250.

[40] Gusev, M. & Evans, D.J. (1992). *Nonlinear Transformations of the Matrix Multiplication Algorithm*, Computer Math.45, pp. 1-21.

[41] Gusev, M. & Evans, D.J. (1993). *The Fastest Matrix Vector Multiplication*, Parallel Algorithms and Applications, Vol. 1, pp. 57-67.

[42] Hansen, P. B. (1975). *The Programming Lanquage Concurrent Pascal*, IEEE Trans. on Software Engineering SE-1, 2 (June), pp 199-206.

[43] Hartsfield, N. & Ringel, G. (1994). *Pearls in Graph Theory : A Comprehensive Introduction*, Academic Press, U.S.A.

[44] Hwang, K. & Xu, Z. (1988). *Multipipeline Networking for Compound Vector Processing,* IEEE Transactions on Computers, v. 37, No. 1.

[45] Hwang, K. (1993). *Advanced Computer Architecture : Parallelism, Scalability, Programmability*, Singapore : McGraw-Hill.

[46] Jagadish, H.V., Rao, S.K. & Kailth, T. (1987). *Array Architectures for Iterative Algorithms*, Proc. of the IEEE 75 (9), pp. 1304-1321.

[47] Kung, H.T. (1980). *Introduction to VLSI*, Mead & Conway, Chapter 8, Addison & Wesley.

[48] Kung, S.Y. (1988). *VLSI Array Processors*, New Jersey: Prentice Hall.

[49] Lee, H.B. & Grondin, R.O. (1988). *A Comparison of Systolic Architectures for Matrix Multiplication*, IEEE Solid-State Circ. 23 (1), pp. 285-289.

[50] Leiserson, C.E. (1981). *Area Efficient VLSI Computation*, Ph.D. Thesis, CMU.

[51] Lengauer, C. (1991). *A View of Systolic Design*, Proc. of the Int. Conf. Parallel Computing Technologies, Novisibirsk, pp. 32-46, World Scientific, Singapore.

[52] Lester, B. (1988). *The Art of Parallel Programming,* Prentice-Hall.

[53] Li, G.J. & Wah, B.W. (1985). *The Design of Optimal Systolic Arrays*, IEEE Trans. Comput. C-34, pp. 66-77.

[54] Liskov, B.L. & Scheifler, R. (1982*). Guardians and Actions : Linguistic Support for Robust, Distributed Programs*, Procs of 9th ACM Symbosium on Reliability in Distributed Software and Database Systems (July), IEEE, New York, pp. 53-60.

[55] Mai, S.W. & Evans, D.J. (1984). *A General Strategy on the Bandwidth Minimization Problem*, Int, J. Computer Math., Vol. 15, pp. 319-337.

[56] Megson, G.M. (1987). *Novel Algorithms for the Soft Systolic Paradigm*, Ph.D. Thesis, L.U.T., U.K.

[57] Milentijevic, I.Z., Milovanovic, I.Z., Milovanovic, E.I. & Stojcev, M.K. (1997). *The Design of Optimal Planar Systolic Arrays for Matrix Multiplication*, Computers Math. Applic.33 (6), pp. 17-35.

[58] Miranker, W.L. & Winkler, A. (1984). *Space Time Representations of Computational Structures*, Computing 32, pp. 93-114.

[59] Moldovan, D.I. & Fortes, J.A.B. (1986). *Partitioning of Algorithms for Fixed Size VLSI Architectures*, IEEE Trans. Comput.C-35 (1), pp. 1-12.

[60] Moldovan, D.I. (1983). *On the Design of Algorithms for VLSI Systolic Arrays*, *Proc. of the IEEE 71*, pp. 113-120.

[61] Moldovan, D.I. (1993). *Parallel Processing: From Applications to Systems*, Morgan Kaufmann, San Mateo, CA.

[62] Myoupro, J.F. & Fabret, A.C., *Designing Modular Linear Systolic Arrays Using Dependence Graph Regular Partitions*, Parallel Processing Letters (to appear).

[63] Rosen, R. (1968). *Matrix Bandwidth Minimization*, Proceedings of the 23rd National Conference, Association for Computer Machinery, Brandon Systems Press, Princeton, N.J.

[64] Sedukhin, S.G. & Karpetian, G.Z. (1990). *Design of Optimal Systolic Systems for Matrix Multiplication of Different Structures*, (in Russian), Report 885, Comp. Center, Siberian Division of USSR Academy of Science, Novosibirsk.

[65] Sedukhin, S.G. (1991). *The Designing and Analysis of Systolic Algorithms and Structures*, (in Russian), *Programming* (2), pp. 20-40.

[66] Shang, W. & Fortes, J.A.B. (1992). *On Time Mapping of Uniform Dependence Algorithms into Lower Dimensional Processor Arrays*, *IEEE Trans. Parallel Distributed Systems 3*(3), pp. 350-363.

[67] Uhr, L. (1984). *Algorithm-Structured Computer Arrays and Networks*, Orlando:Academic Press.

[68] United States Department of Defence (1981). *Programming Lanquage Ada : Reference Manual*, v. 106, Lecture Notes in Computer Science, New York : Springer Verlag.

[69] Wilson, R.J. & Watkins, J.J. (1990). *Graphs: An introductory approach*, New York: Wiley.

[70] Wirth, N. (1977). *Modula : A Lanquage for Modular Multiprogramming*, Software Practice and Experience 7, pp 33-35.

[71] Μπεκάκος, Μ.Π. (1993). *Αποτίμηση και Πρόβλεψη Απόδοσης Συστημάτων Υπολογιστών*, Εκδόσεις Α. Σταμούλης, Αθήνα.

[72] Μπεκάκος, Μ.Π. (1993). *Αρχιτεκτονική Υπολογιστών και Τεχνολογία Παράλληλης Επεξεργασίας*, Τόμος ΙΙ, Εκδόσεις Α. Σταμούλης, Αθήνα.

[73] Μπεκάκος, Μ.Π. (1994). *Αρχιτεκτονική Υπολογιστών και Τεχνολογία Παράλληλης Επεξεργασίας*, Τόμος Ι, Εκδόσεις Α. Σταμούλης, Αθήνα.

[74] Μπεκάκος, Μ.Π. (1997). *Αρχιτεκτονική Υπολογιστών και Τεχνολογία Παράλληλης Επεξεργασίας*, Τόμος ΙΙΙ, Εκδόσεις Α. Σταμούλης, Αθήνα.

Σημειώσεις - Παρατηρήσεις - Σχόλια