ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΟΜΕΑΣ ΘΕΩΡΗΤΙΚΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Παράλληλοι Αριθμητικοί Αλγόριθμοι

Σημειώσεις

Διδάσκων : Φ. ΤΖΑΦΕΡΗΣ

ΑΘΗΝΑ, 2010

Παράλληλοι Υπολογισμοί Πινάκων

- Αναστροφή Πίνακα : $\mathbf{A} \longrightarrow \mathbf{A^T}$

- Πολ/σμός Πινάκων : $(\mathbf{A}, \mathbf{B}) \longrightarrow \mathbf{C} = \mathbf{AB}$

- Πολ/σμός Πίνακα με διάνυσμα :
$$(\mathbf{A}, \mathbf{u}) \longrightarrow \mathbf{v} = \mathbf{Au}$$

Δίκτυα Ενδοεπικοινωνίας Παραλλήλων Υπολογιστών

- Μονοδιάστατος πίνακας (**linear array**)

- Ορθογώνιο πλέγμα (**mesh**)

- Δένδρο (**tree**)

- Τέλειο μοίρασμα (**perfect shuffle**)

- Υπερκύβος (**hypercube**)

Απεικόνιση(**mapping**) Πινάκων στους επεξεργαστές: επηρεάζει σημαντικά την επίδοση ενός παράλληλου συστήματος.

Αραιός: λέγεται ένας πίνακας όταν ένα σημαντικό ποσοστό των στοιχείων του (συνήθως $> 80\%$) είναι μηδενικά.

Πυκνός: είναι ένας πίνακας που δεν είναι αραιός.

1. Αναστροφή πίνακα

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \end{array} \longrightarrow A^T = \begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} & \alpha_{41} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} & \alpha_{42} \\ \alpha_{13} & \alpha_{23} & \alpha_{33} & \alpha_{43} \\ \alpha_{14} & \alpha_{24} & \alpha_{34} & \alpha_{44} \end{bmatrix}$$

$\alpha_{i,j}$: ακέραιος, πραγματικός, χαρακτήρας, κ.α.

Ακολουθιακή αναστροφή

procedure **TRANSPOSE(A)**

for $i = 2$ **to** n **do**

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

    $\alpha_{i,j} \longleftrightarrow \alpha_{j,i}$

  end for

end for

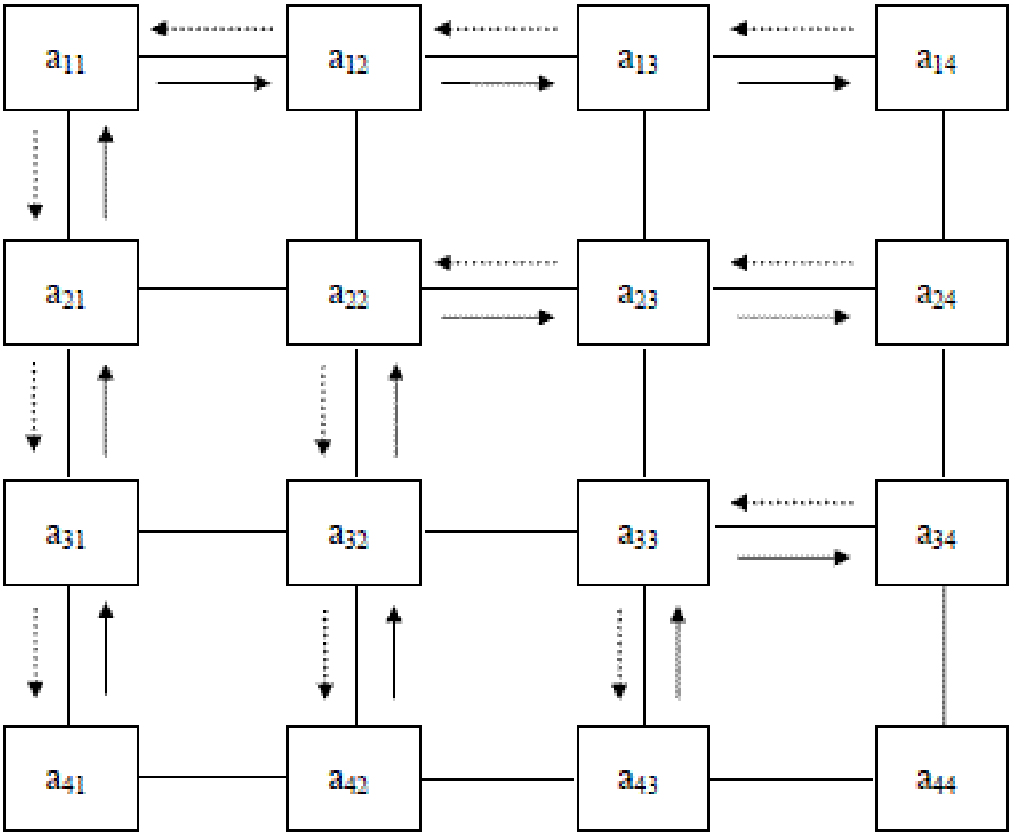
Παράλληλη αναστροφή

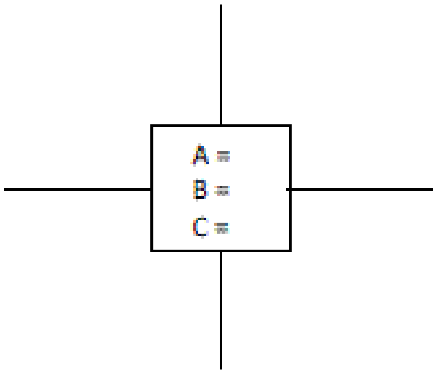
Μοντέλα παράλληλου υπολογισμού

- ορθογώνιο πλέγμα (**mesh**)
- **shuffle** (μοίρασμα)
- **shared-memory SIMD**

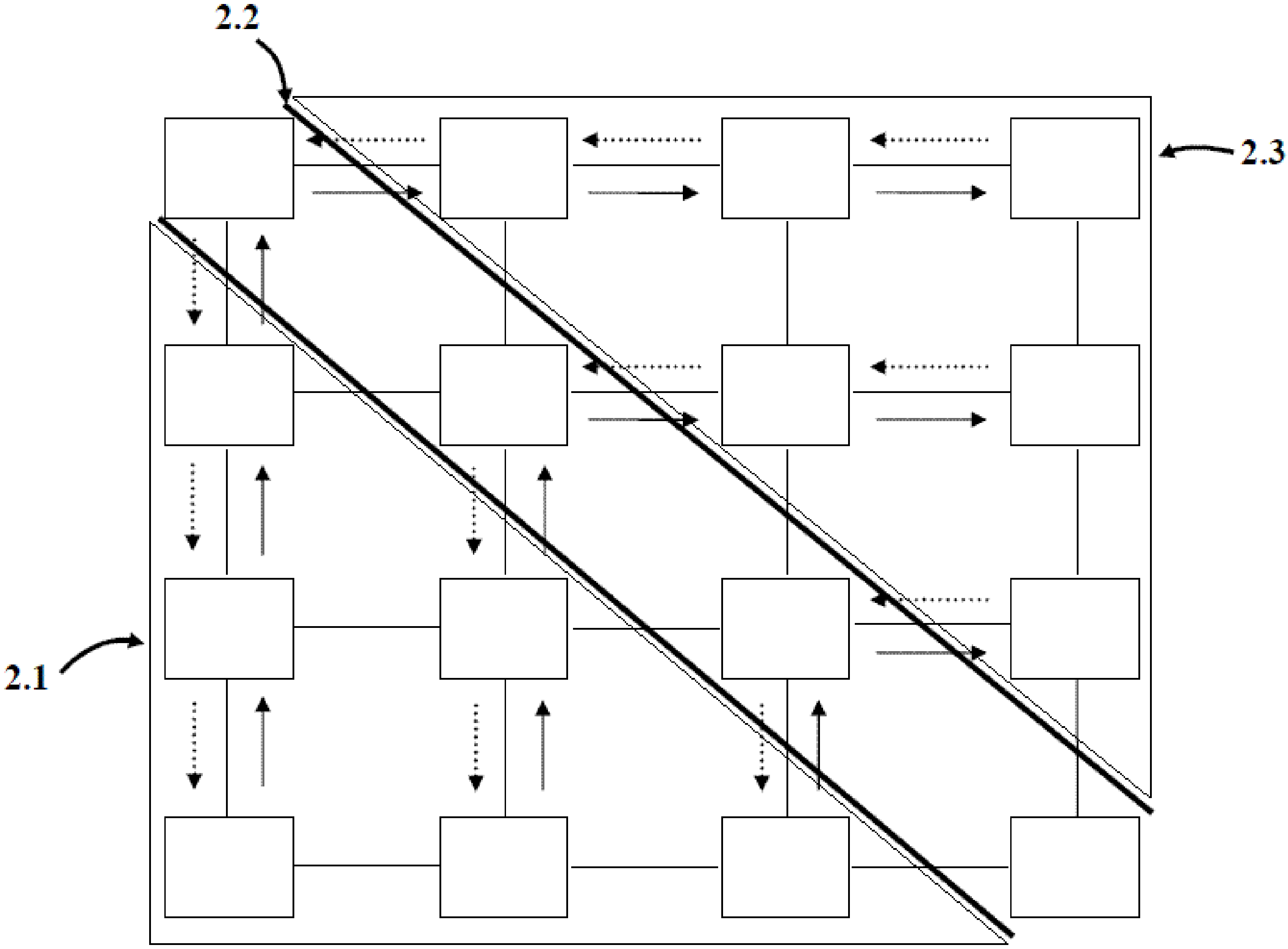
1.1 Αναστροφή πίνακα σε δίκτυο (mesh)

Ένας πίνακας A για να αναστραφεί καταχωρείται σε ένα δίκτυο (mesh)





A, B, C <u>καταχωρητές</u>

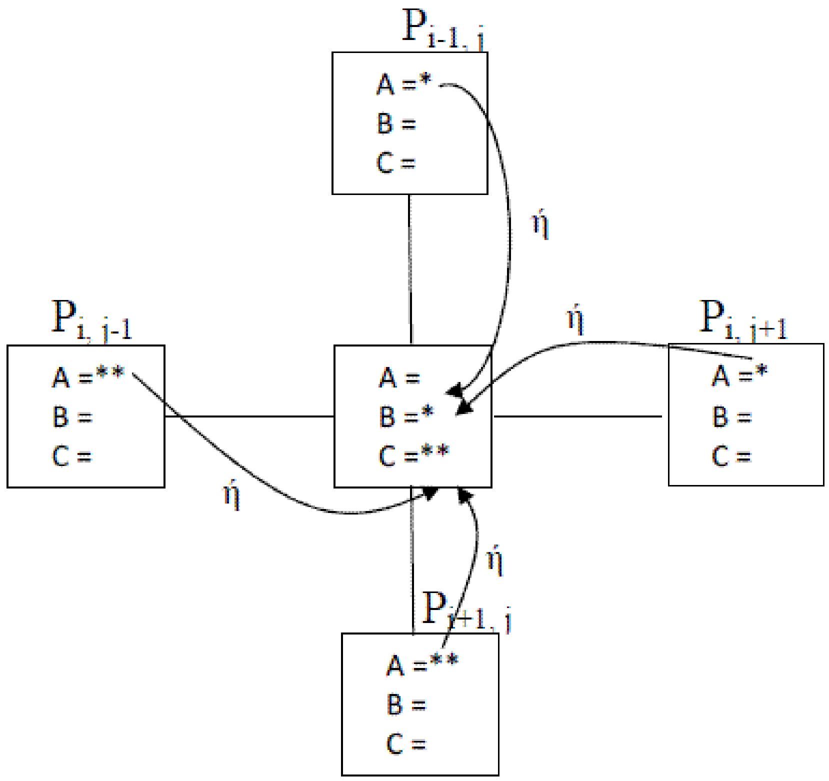


$(\alpha_{km}, \underbrace{m, k})$: η μεταβιβαζόμενη πληροφορία

στοιχείο   προορισμός (m, k)

κάτω φράγμα χρόνου εκτέλεσης

$\underline{\Omega(n) = 2n - 2}$



Κάθε επεξεργαστής P_{ij} έχει 3 καταχωρητές

1. A_{ij}: χρειάζεται για να καταχωρείται αρχικά το α_{ij} και τελικά το α_{ji}

2. B_{ij}: χρειάζεται για την καταχώρηση των δεδομένων που δέχεται από τον γειτονικό του δεξιά $P_{i,j+1}$ ή τον γειτονικό του πάνω $P_{i-1,j}$

3. C_{ij}: χρειάζεται για την καταχώρηση των δεδομένων που δέχεται από τον γειτονικό του αριστερά $P_{i,j-1}$ ή τον γειτονικό του κάτω $P_{i+1,j}$

Ανάλυση

Κάθε στοιχείο α_{ij} $(i > j)$
που βρίσκεται κάτω από τη κύρια διαγώνιο
διασχίζει προς τα πάνω τη στήλη του μέχρι
να φθάσει στον P_{jj} και στη συνέχεια διασχίζει
τη γραμμή j μέχρι να φθάσει στον P_{ji}.

Κάθε στοιχείο α_{ij} $(j > i)$
που βρίσκεται πάνω από τη κύρια διαγώνιο
διασχίζει προς τα αριστερά τη γραμμή του μέχρι
να φθάσει στον P_{ii} και στη συνέχεια διασχίζει
τη στήλη i μέχρι να φθάσει στον P_{ji}.

Το μακρύτερο μονοπάτι είναι εκείνο που διασχίζει
το α_{n1} (ή α_{1n}) που απαιτεί

$2(n-1)$ βήματα

<u>χρόνος εκτέλεσης</u>: $t(n) = O(n)$

<u>αριθμός επεξεργαστών</u>: $p(n) = n^2$

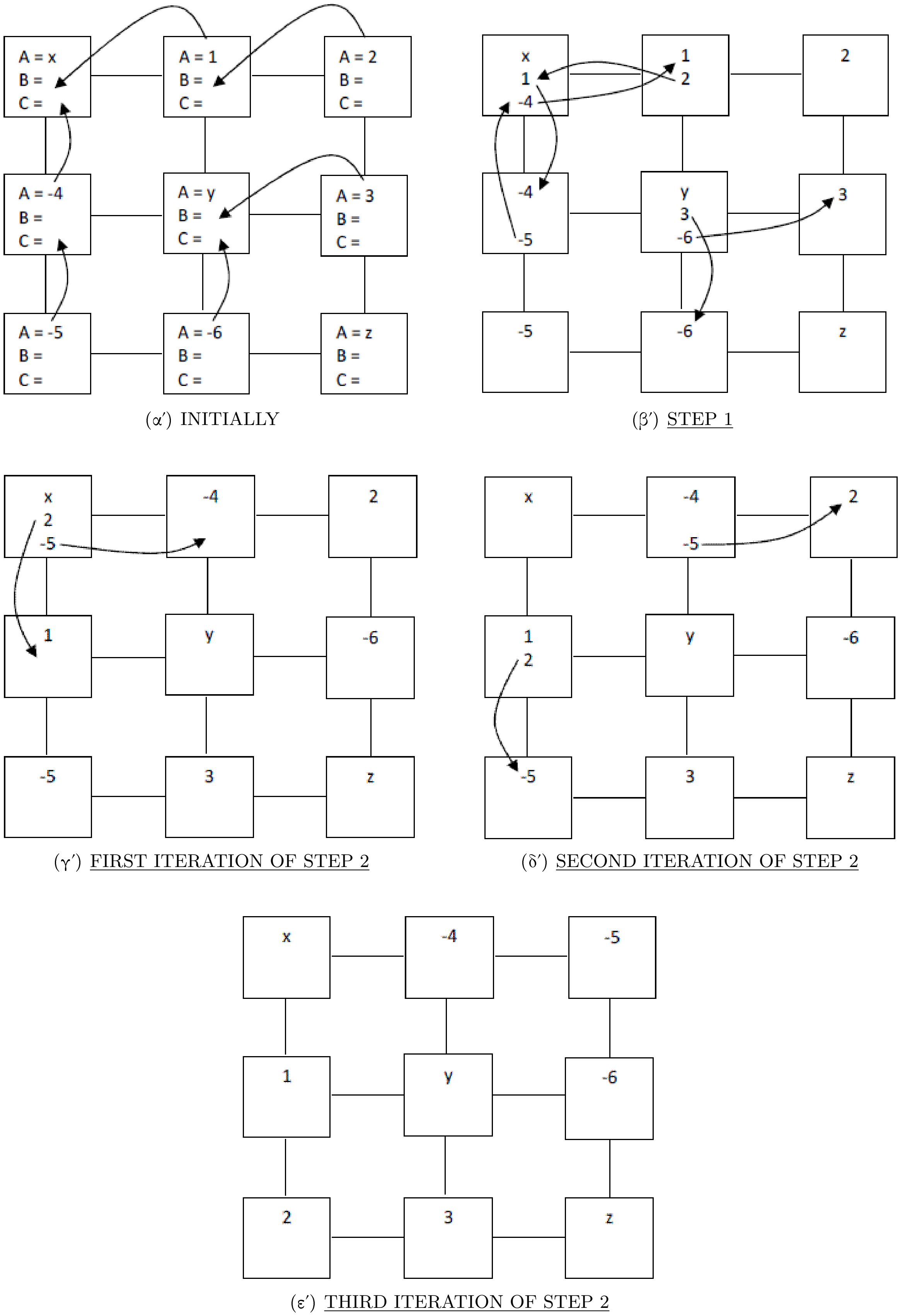
<u>κόστος</u>: $O(n^3)$ (όχι βέλτιστος)

procedure MESH TRANSPOSE(A)

```
Step 1:     do steps 1.1 and 1.2 in parallel

            (1.1) for i = 2 to n do in parallel
        C               for j = 1 to i – 1 do in parallel
                            C(i - 1, j) ← (a_ij, j, i)
                        end for
        A             end for
            (1.2) for i = 1 to n – 1 do in parallel
             A          for j = i + 1 to n do in parallel
       B                    B(i, j - 1) ← (a_ij, j, i)
                        end for
                      end for.

Step 2:     do steps 2.1, 2.2, and 2.3 in parallel

            (2.1) for i = 2 to n do in parallel
                        for j = 1 to i – 1 do in parallel
                            while P(i, j) receives input from its neighbors do
                              (i)  if (a_km, m, k) is received from P(i + 1, j)
                                   then send it to P(i - 1, j)
                                   end if
                              (ii) if (a_km, m, k) is received from P(i - 1, j)
                                   then if i = m and j = k
                                        then A(i, j) ← a_km {a_km has reached its destination}
                                        else send (a_km, m, k) to P(i + 1, j)
                                        end if
                                   end if
                            end while
                        end for
                  end for
```

(2.2) **for** i = 1 **to** n **do** in parallel
  while P(i, i) receives input from its neighbors **do**
    (i) **if** (a_{km}, m, k) is received from P(i + 1, i)
      then send it to P(i, i + 1)
      end if
    (ii) **if** (a_{km}, m, k) is received from P(i, i + 1)
      then send it to P(i, i + 1)
      end if
  end while
end for

(2.3) **for** i = 1 **to** n - 1 **do** in parallel
  for j = i + 1 **to** n **do** in parallel
    while P(i, j) receives input from its neighbors **do**
      (i) **if** (a_{km}, m, k) is received from P(i, j + 1)
        then send it to P(i, j - 1)
        end if
      (ii) **if** (a_{km}, m, k) is received from P(i, j - 1)
        then if i = m **and** j = k
          then A(i, j) $\leftarrow$ a_{km} {a_{km} has reached its destination}
          else send (a_{km}, m, k) to P(i, j + 1)
          end if
        end if
    end while
  end for
end for. □



(α′) INITIALLY

(β′) STEP 1

(γ′) FIRST ITERATION OF STEP 2

(δ′) SECOND ITERATION OF STEP 2

(ε′) THIRD ITERATION OF STEP 2

Σχήμα 1: Transporting matrix using procedure MESH TRANSPOSE

1.2 Αναστροφή πίνακα με <u>**Shuffle**</u>.

A $\quad n \times n$

$n = 2^q$

<u>πλήθος επεξεργαστών</u>: n^2

<u>Επεξεργαστές</u>: $P_0, P_1, ..., P_{2^{2q}-1}$

$\alpha_{ij} \quad \xrightarrow{\underline{\text{καταχώρηση}}} \quad P_K$

$$K = 2^q(i-1) + (j-1)$$

<u>Παράδειγμα</u>

$\underline{q=2},\ \underline{n = 2^2 = 4},\ p(n) = 16$

$\alpha_{33} \longrightarrow P_{10}$

γιατί για $i = 3,\ j = 3 \Longrightarrow K = 2^2(3-1) + (3-1) = 10$

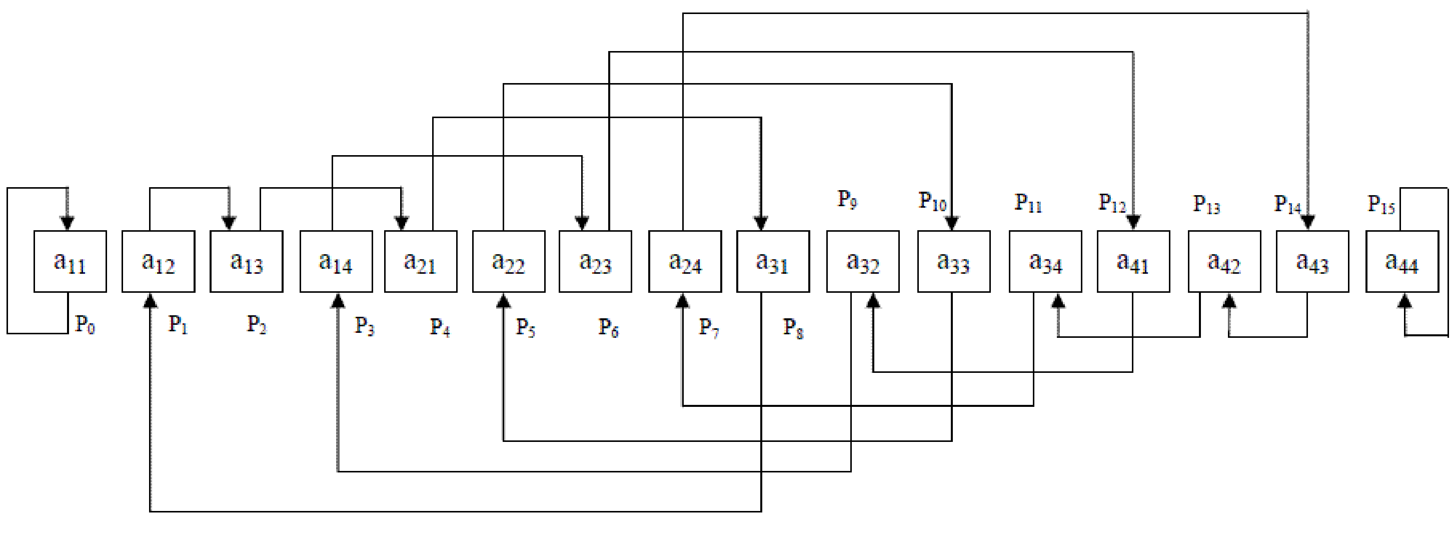
Μετά από <u>q ακριβώς **shuffle** πράξεις</u> (μοιράσματα) ο επεξεργαστής P_K θα περιέχει το στοιχείο α_{ji}.

Για να δούμε αυτό υποθέτουμε ότι:

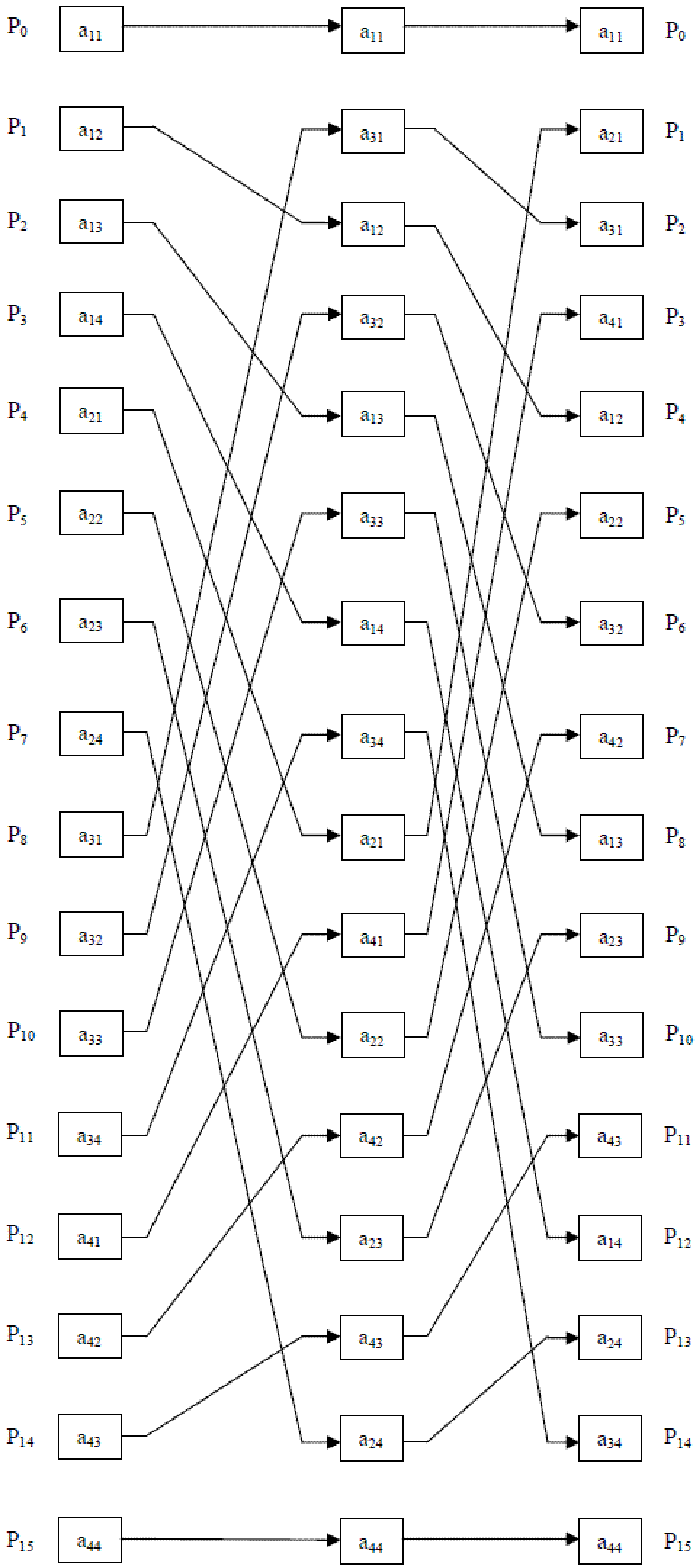
Αν ο P_K συνδέεται με τον P_m, τότε ο m προκύπτει από τον K με μια ολίσθηση κυκλικά προς τα αριστερά κατά μια θέση της δυαδικής παράστασης του K.

Στο παραπάνω παράδειγμα, όπου $q = 2$ έχουμε τις ακόλουθες συνδέσεις μεταξύ των $2^4 = 16$ επεξεργαστών.

Ο επεξεργαστής με δείκτη k		συνδέεται →		Με τον επεξεργαστή με δείκτη m
0	0000		0000	0
1	0001		0010	2
2	0010		0100	4
3	0011		0110	6
4	0100		1000	8
5	0101		1010	10
6	0110	Κυκλική	1100	12
7	0111	→ ολίσθηση κατά 1	1110	14
8	1000	προς τα αριστερά	0001	1
9	1001		0011	3
10	1010		0101	5
11	1011		0111	7
12	1100		1001	9
13	1101		1011	11
14	1110		1101	13
15	1111		1111	15



Transposition



Έστω ότι ο δείκτης K ενός επεξεργαστή αποτελείται από $2q$ **bits**.

Αν $\underline{K = 2^q(i-1) + (j-1)}$ τότε

$\underline{q}$ τα πλέον σημαντικά **bits** αναπαριστούν τον $\underline{i-1}$

αφού $\underline{q}$ τα λιγότερο $\gg$ $\gg$ $\gg$ $\underline{j-1}$

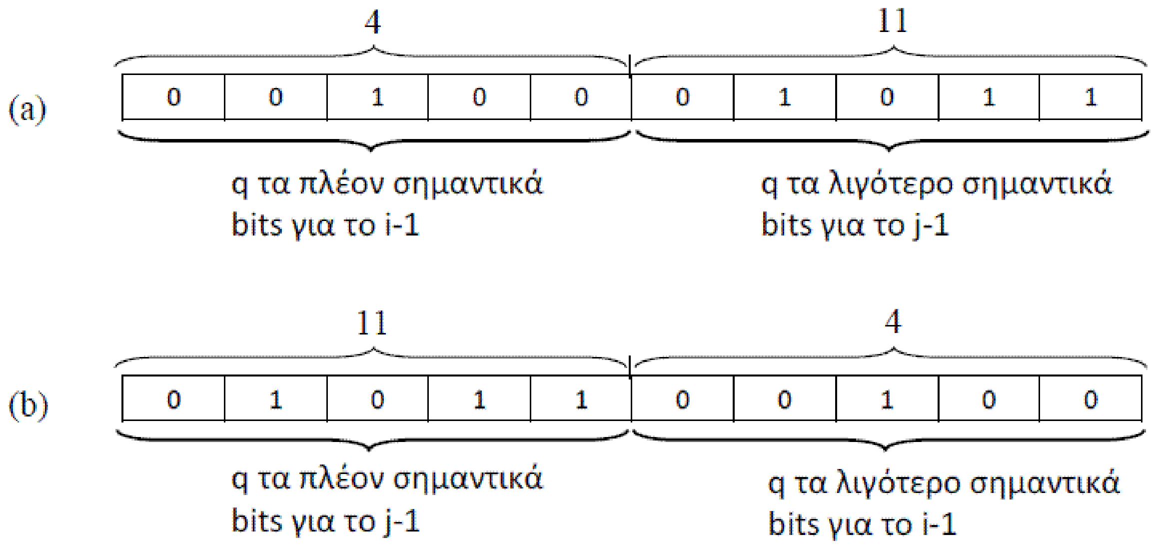
Παράδειγμα $\quad q = 5,\ i = 5,\ j = 12$

$$K = 2^5(5-1) + (12-1) = 128 + 11 = 139$$

Μετά από q **shuffles** (κυκλικές ολισθήσεις προς τα αριστερά) στοιχείο α_{ij} που κατείχε αρχικά ο P_K θα βρίσκεται στον επεξεργαστή P_S με δείκτη

$$\underline{S = 2^q(j-1) + (i-1)}$$

Δηλαδή το στοιχείο α_{ij} έχει μετακινηθεί στη θέση που κατείχε αρχικά το α_{ji}



procedure <u>SHUFFLE TRANSPOSE(A)</u>

```
for i = 1 to q do
    for k = 1 to 2^{2q}-1 do in parallel
        ο P_K στέλνει το στοιχείο του A που κατέχει
        στον P_{2K mod (2^{2q} - 1)}
    end for
end for.
```

Το **procedure** αυτό εκτελεί q επαναλήψεις σταθερού χρόνου και επομένως ο χρόνος εκτέλεσης είναι

$$t(n) = O(\log(n))$$

Αφού

$p(n) = n^2$ το υπολογιστικό κόστος είναι

$$C(n) = O(n^2 \log(n))$$

το οποίο δεν είναι βέλτιστο.

Η ενδοεπικοινωνία **shuffle** είναι ταχύτερη από εκείνη στο ορθογώνιο πλέγμα.

1.3 Αναστροφή πίνακα σε ένα EREW SM SIMD υπολογιστή

$A \longrightarrow$ κοινή μνήμη

Επεξεργαστές P_{ij}, $2 \leq i \leq n$

$1 \leq j \leq i-1$

Αριθμός επεξεργαστών: $P(n) = (n^2 - n)/2$

Με όλους τους επεξεργαστές να λειτουργούν παράλληλα ο επεξεργαστής P_{ij} ανταλλάσει τα στοιχεία α_{ij} και α_{ji} του A.

procedure EREW TRANSPOSE(A)

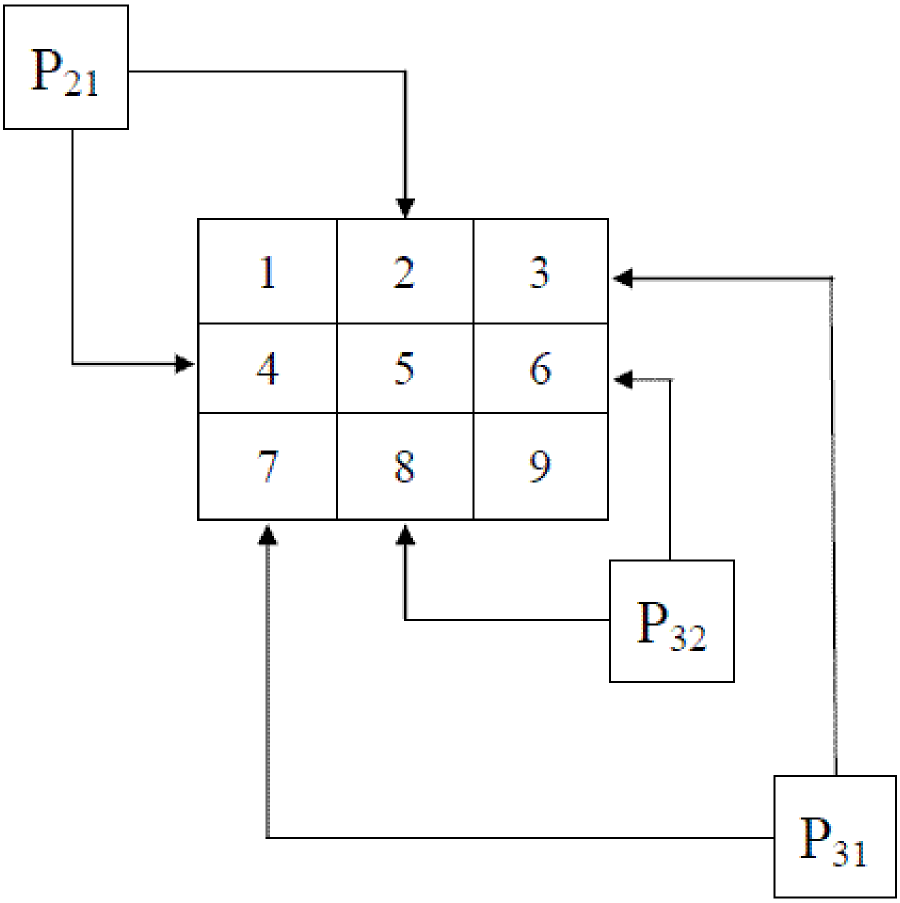
for $i = 2$ **to** n **do** in parallel
  for $j = 1$ to $i - 1$ **do** in parallel
    $\alpha_{ij} \longleftrightarrow \alpha_{ji}$
  end for
end for.

Απαιτείται σταθερός χρόνος για να ανταλλαγούν δύο στοιχεία στον κάθε επεξεργαστή.

Χρόνος εκτέλεσης $t(n) = O(1)$

Αφού $P(n) = O(n^2)$ το υπολογιστικό κόστος είναι

$C(n) = O(n^2)$ που είναι βέλτιστο.



Αναστροφή πίνακα $n = 3$

με το procedure <u>EREW TRANSPOSE(A)</u>