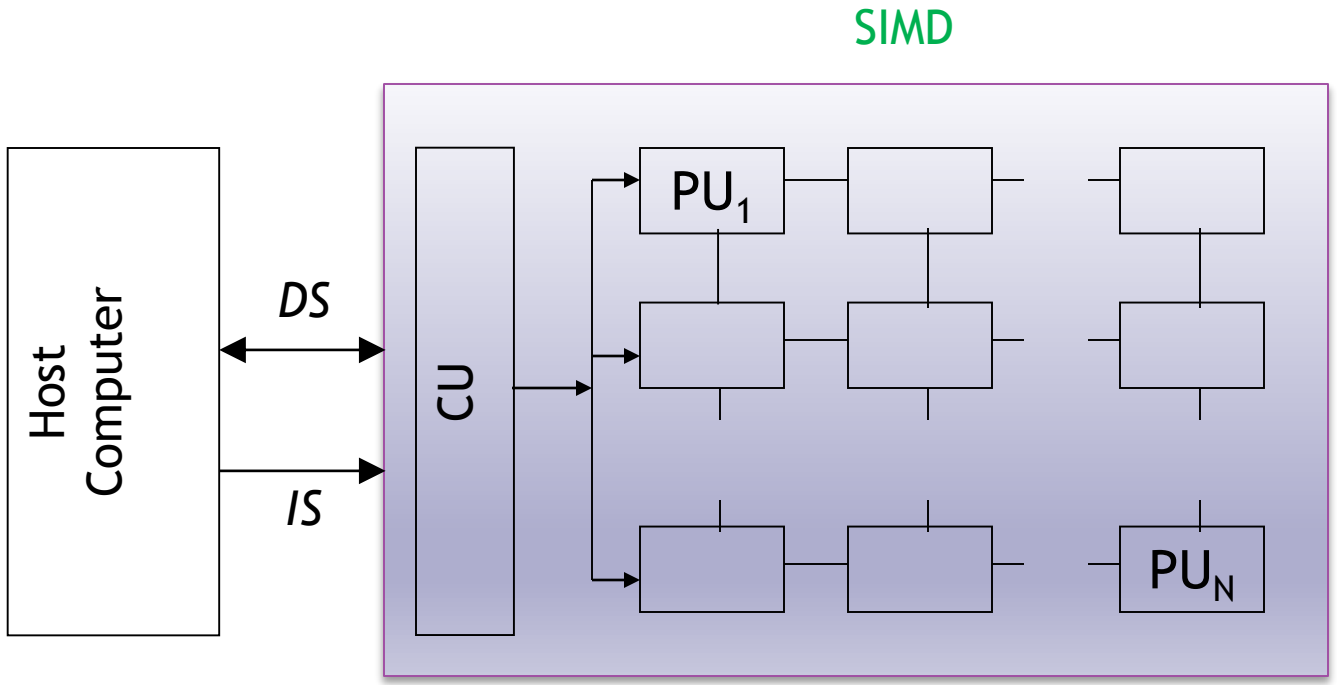
Προηγμένες αρχιτεκτονικές υπολογιστών και προγραμματισμός παραλλήλων συστημάτων

12. Συστολικές συστοιχίες επεξεργαστών

Συστολικές συστοιχίες επεξεργαστών (ΣΕ)

- Γνωστές και ως systolic arrays.
- Αποτελούν παράδειγμα μηχανών SIMD (Single-Instruction-Multiple-Data) που εκτελούν συγχρονισμένα πιο πολύπλοκους αλγορίθμους από τους διανυσματικούς υπολογιστές.
  - Πχ. Συνέλιξη δύο χρονοσειρών, μετασχηματισμός Fourier, κλπ.
- Εκτελούν κάποιες πολύ συγκεκριμένες εργασίες με τον βέλτιστο παράλληλο τρόπο.

Σχεδιάγραμμα ΣΕ



Χαρακτηριστικά ΣΕ

- Προσκολλημένοι (attached) σε έναν υπολογιστή host. Δεν στέκουν αυτόνομα.
- Όχι λειτουργικό σύστημα.
- Σύγχρονη επικοινωνία.
- Όλοι οι επεξεργαστές εκτελούν την ίδια εντολή σε διαφορετικά δεδομένα (SIMD).
- Εκτελούν ειδικούς αλγορίθμους (special purpose machines).
- Εξειδικευμένοι επεξεργαστές.
- Υψηλό κόστος.

Απεικόνιση αλγορίθμου σε ΣΕ

- Βήματα:
  1. Μετατροπή κώδικα σε κώδικα μοναδικής ανάθεσης (single assignment code).
  2. Σχεδίαση Γράφου Εξάρτησης (Γ.Ε.).
  3. Μετατροπή μακρινών ακμών σε τοπικές.
  4. Προβολή Γ.Ε. σε Γράφο Ροής Σήματος.
  5. Χρονοδιάγραμμα.
  6. Σχεδίαση Συστολικού Επεξεργαστή.

Παράδειγμα: Πολλαπλασιασμός πίνακα επί διάνυσμα

- A = πίνακας $m \times n$
- b = διάνυσμα μήκους n (ισοδύναμος πίνακας $n \times 1$).
- c = αποτέλεσμα $A \cdot b$ = διάνυσμα μήκους m (ισοδύναμος πίνακας $m \times 1$).
- $c(i) = \sum_{j=1}^{n} A(i,j) \cdot b(j)$.
- $i = 1,2,\ldots,m$

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

$$\begin{aligned} \mathbf{Ax} &= \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 \cdot 1 - 1 \cdot 1 + 0 \cdot 2 \\ 2 \cdot 0 - 1 \cdot 3 + 0 \cdot 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ -3 \end{bmatrix}. \end{aligned}$$

Πίνακας επί διάνυσμα: Σειριακός κώδικάς

```
for (i = 1 to m) {
    c[i] = 0;
    for (j = 1 to n) {
        c[i] = c[i] + A[i][j] * b[j];
    }
}
```

Έλεγχος κώδικα μοναδικής ανάθεσης

- Έστω m=4, n=3. Loop unrolling:

```
i=1: c[1]=0;
     c[1]=c[1] + A[1][1]*b[1];
     c[1]=c[1] + A[1][2]*b[2];
     c[1]=c[1] + A[1][3]*b[3];
i=2: c[2]=0;
     c[2]=c[2] + A[2][1]*b[1];
     c[2]=c[2] + A[2][2]*b[2];
     ...
```

4 αναθέσεις στο `c[1]`

Άρα: OXI single assignment code

```
for (i = 1 to m) {
   c[i] = 0;
   for (j = 1 to n) {
      c[i] = c[i] + A[i][j] * b[j];
   }
}
```

Κώδικας μοναδικής ανάθεσης (1)

- Τροποποιούμε τον κώδικα έτσι ώστε σε κάθε μεταβλητή να γίνεται ανάθεση το πολύ μία φορά.
- Αρίθμηση αναθέσεων:

```
i=1:  c[1][0] = 0;
      c[1][1] = c[1][0] + A[1][1]*b[1];
      c[1][2] = c[1][1] + A[1][2]*b[2];
      c[1][3] = c[1][2] + A[1][3]*b[3];
i=2:  c[2][0] = 0;
      c[2][1] = c[2][0] + A[2][1]*b[1];
      c[2][2] = c[2][1] + A[2][2]*b[2];
      ...
```

Κώδικας μοναδικής ανάθεσης (2)

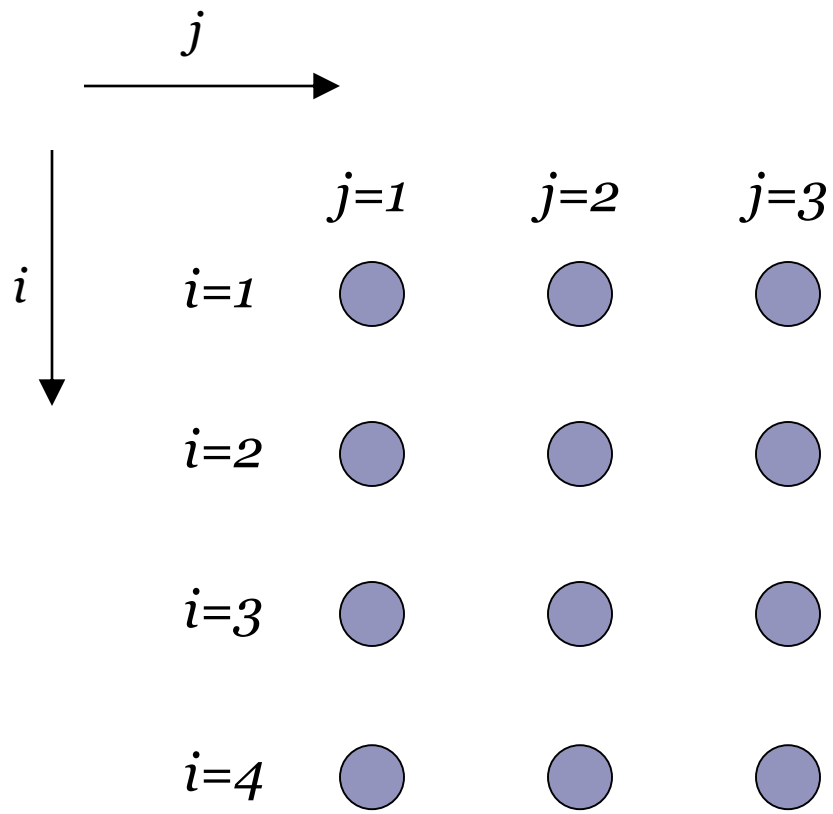
- Ισοδύναμος κώδικας στον οποίο κάθε ανάθεση γίνεται το πολύ μία φορά.
- Θυμηθείτε ότι το αποτέλεσμα του πολλαπλασιασμού είναι διάνυσμα (δηλ. πίνακας με μία στήλη).

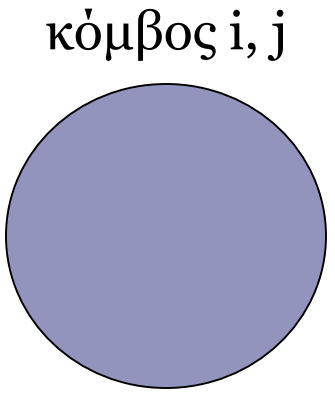
```
for (i = 1 to 4) {
   c[i][0] = 0;
   for (j = 1 to 3) {
      c[i][j] = c[i][j-1] + A[i][j]*b[j];
   }
}
```

Σχεδιασμός γράφου εξάρτησης

- Δύο loop το ένα μέσα στο άλλο.
- Δύο διαστάσεις στο γράφο εξάρτησης:
  - μία διάσταση για το δείκτη i και
  - μία διάσταση για το δείκτη j
- Κάθε κόμβος του γράφου = οι πράξεις που γίνονται κατά το iteration (i,j).
- Κάθε ακμή του γράφου = δεδομένα που ανταλλάσσονται μεταξύ κόμβων.

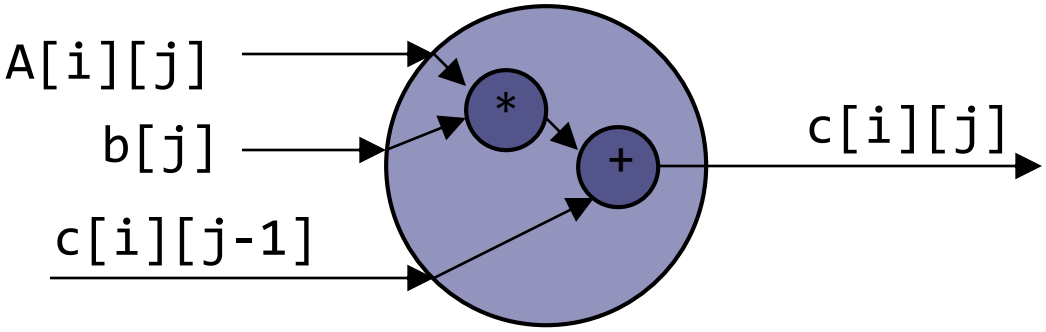
Σχεδιασμός γράφου εξάρτησης (2)





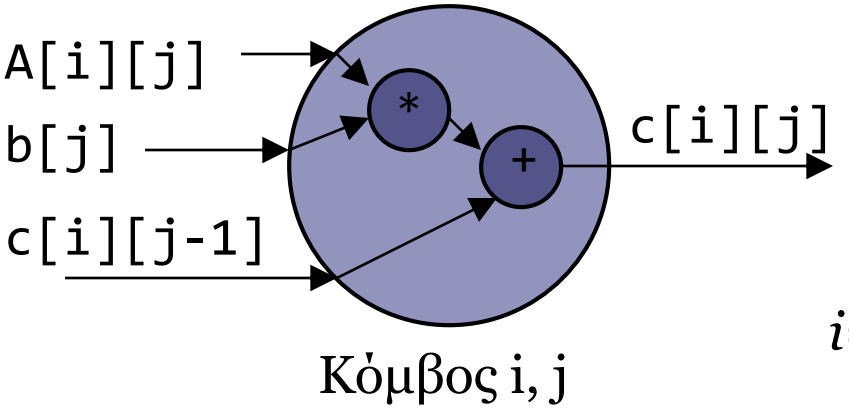
Σχεδιασμός κόμβου γράφου εξάρτησης

```
for (i = 1 to 4) {
   c[i][0] = 0;
   for (j = 1 to 3) {
      c[i][j] = c[i][j-1] + A[i][j]*b[j];
   }
}
```

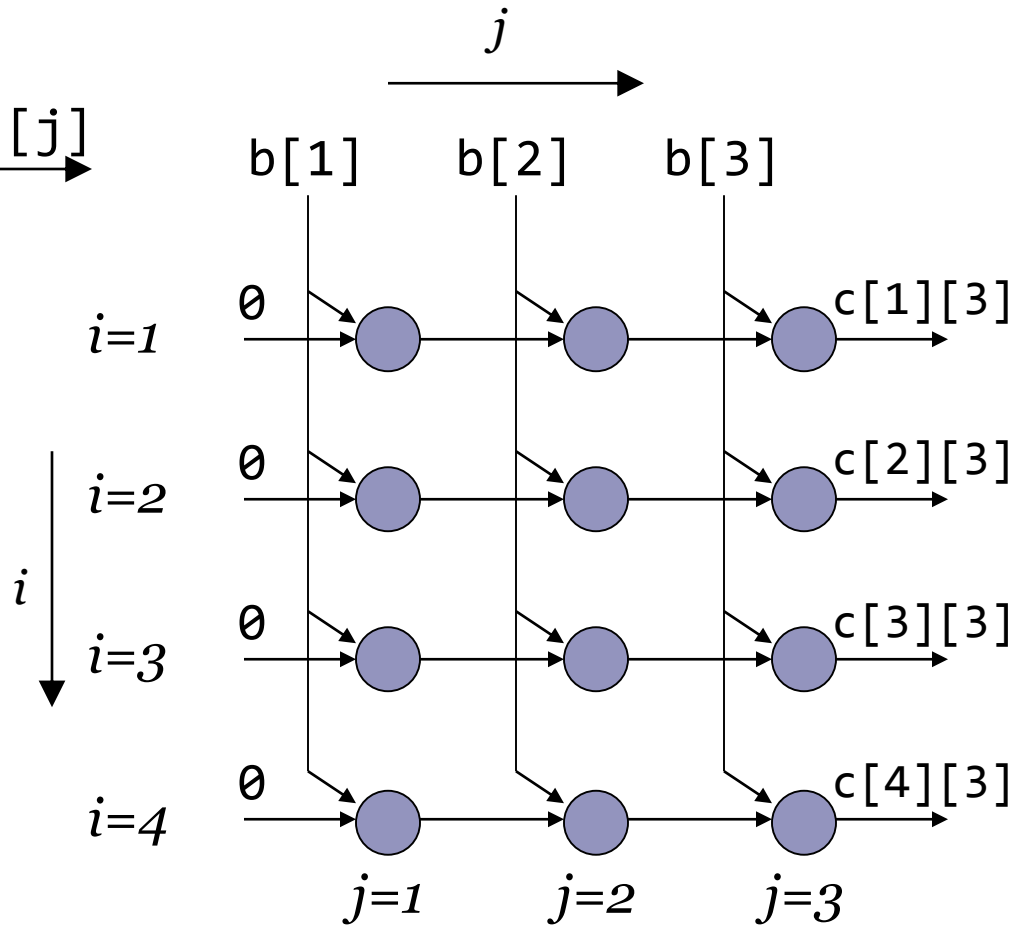


Κόμβος i, j

Σχεδιασμός γράφου εξάρτησης (3)



Κόμβος i, j



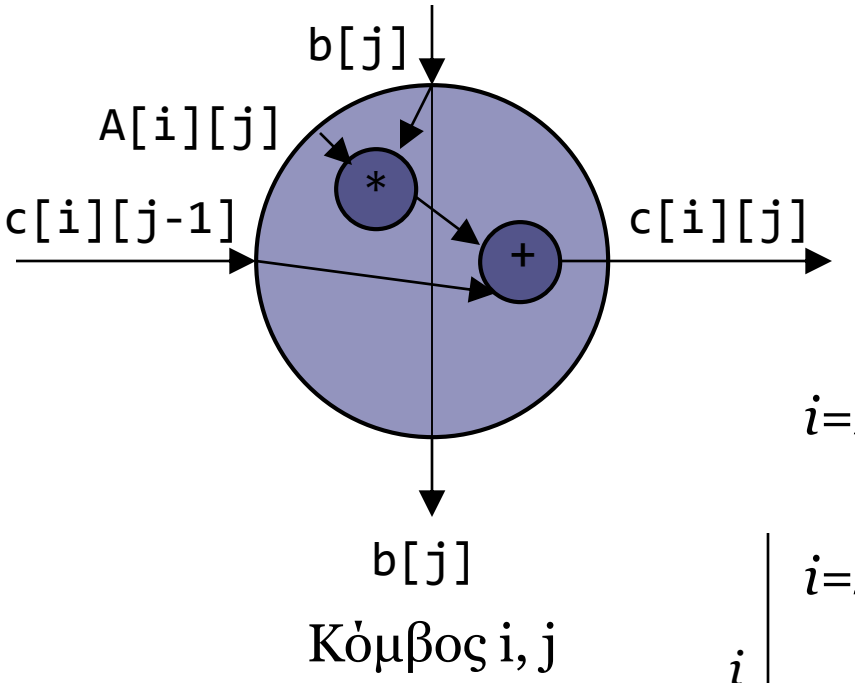
Μετατροπή μακρινών ακμών σε τοπικές

- Οι κόμβοι θα γίνουν επεξεργαστές.
- Οι ακμές θα γίνουν γραμμές επικοινωνίας.
- Οι μακρινές ακμές θα γίνουν bus.
- Ο δίαυλος επικοινωνίας (bus) δεν είναι επιθυμητός στη μαζική παράλληλη επεξεργασία.
- Δημιουργεί πρόβλημα στον συγχρονισμό των επεξεργαστών, ιδίως όταν οι αποστάσεις είναι μεγάλες.

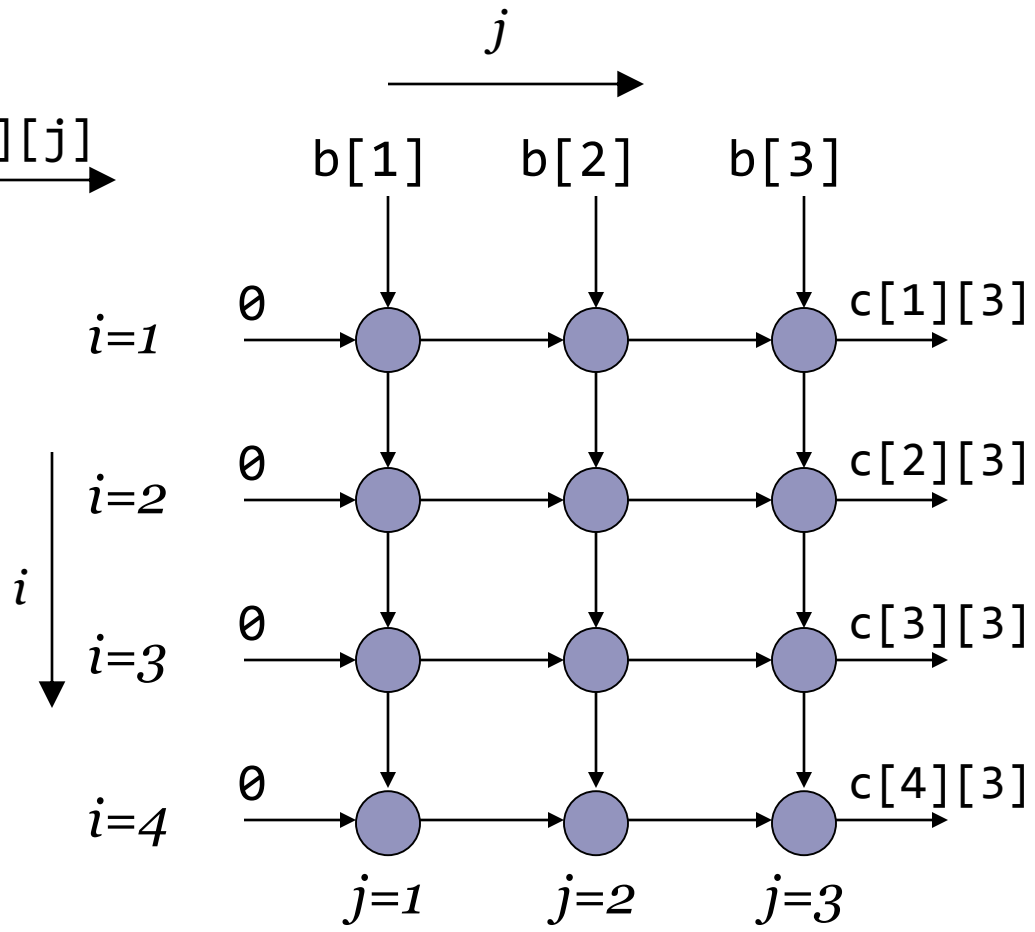
Μακρινές ακμές

- **Παράδειγμα:** Αν η απόσταση από τον πρώτο έως τον τελευταίο κόμβο είναι 1m, το σήμα, που κινείται με την ταχύτητα του φωτός ($3 \cdot 10^8$ m/sec), χρειάζεται $3.3 \cdot 10^{-9}$ = 3.3 nsec για να διανύσει την απόσταση.
- Συγχρονισμένοι κόμβοι: πρέπει να είναι βέβαιο ότι όλοι θα πάρουν τα δεδομένα που απαιτούνται μέσα σε έναν κύκλο ρολογιού.
- Περίοδος ρολογιού διαύλου: μεγαλύτερη από 3.3 nsec.
  - για εξασφάλιση ότι και ο τελευταίος κόμβος έλαβε το δεδομένο.
- Δηλαδή: συχνότητά του διαύλου < 300 MHz.
  - πολύ μικρότερη από την ταχύτητα των επεξεργαστών που συνδέονται στον δίαυλο (πχ. 2-3GHz).
- Για τον λόγο αυτόν, οι μακρινές γραμμές επικοινωνίας αποφεύγονται.

Μετατροπή μακρινών ακμών σε τοπικές (2)



Κόμβος i, j



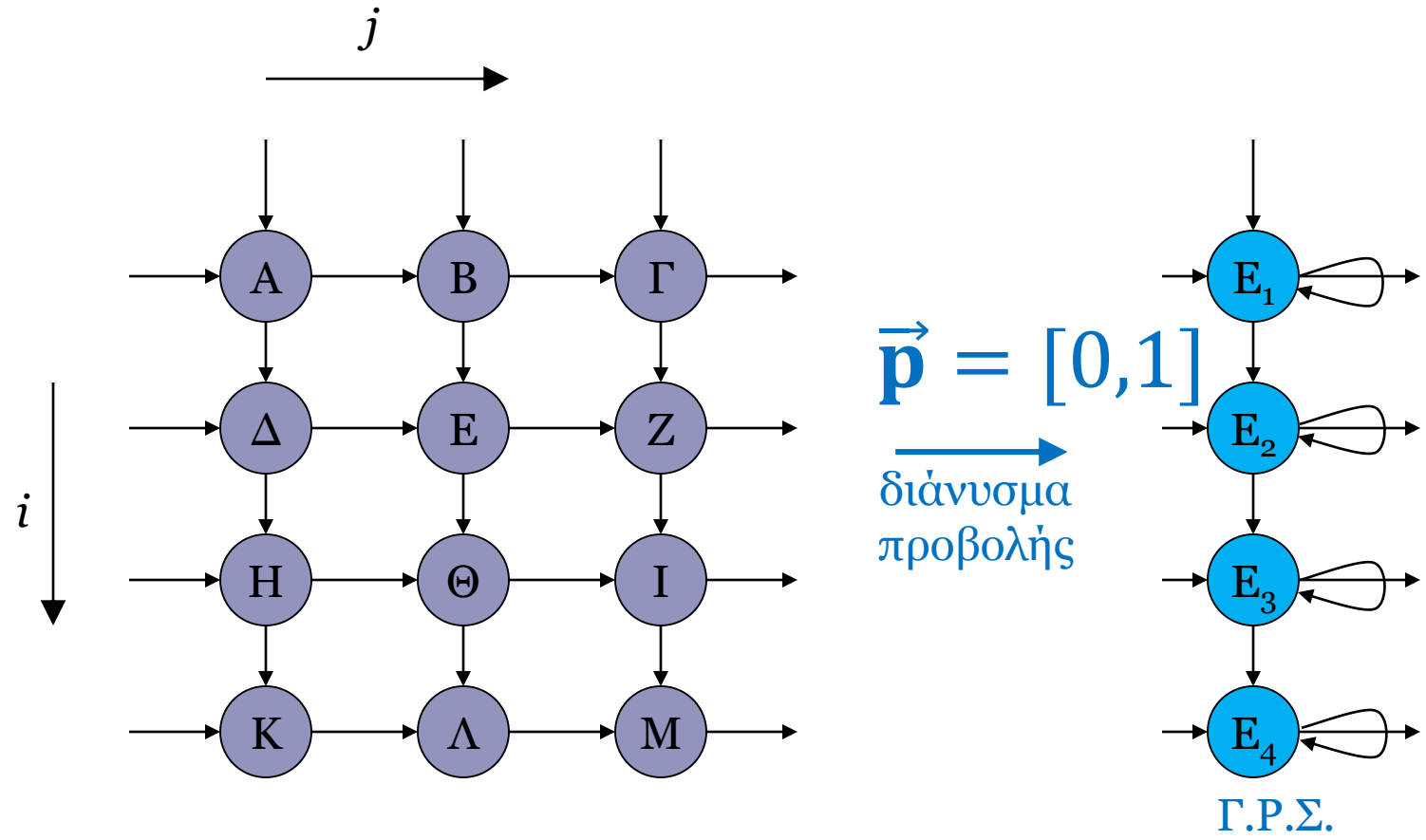
Μετατροπή μακρινών ακμών σε τοπικές (3)

- Η τιμή δεν μεταδίδεται ταυτόχρονα και στους 4 κόμβους της ίδιας σειράς: μεταδίδεται από κόμβο σε κόμβο.
- Δημιουργούνται επιπλέον εξαρτήσεις μεταξύ των κόμβων οι οποίες δεν υπήρχαν στον αρχικό αλγόριθμο.
- Ο μόνος ασφαλής τρόπος για να εξαλείψουμε τις μακρινές συνδέσεις χωρίς επιβάρυνση (τις περισσότερες φορές) στον χρόνο εκτέλεσης του αλγορίθμου.

Γράφος ροής σήματος

- Signal Flow Graph (SFG).
- Μικρότερης διάστασης από το γράφο εξάρτησης.
- Ο γράφος εξάρτησης μπορεί να προβληθεί στο SFG.
- Ορίζεται μία κατεύθυνση – διάνυσμα προβολής $\vec{\mathbf{p}}$.
- Με $\vec{p} = [0 \;\; 1]$ οι πράξεις (κόμβοι) της πρώτης γραμμής (j=1) προβάλλονται στον επεξεργαστή E1, της δεύτερης (j=2) στον E2 κ.ο.κ.
- Οι ακμές του γράφου προβάλλονται επίσης σε ακμές επάνω στο SFG.

Προβολή σε γράφο ροής σήματος



Χρονοδιάγραμμα (1)

- Χρονοδιάγραμμα εκτέλεσης του προγράμματος
- Ορισμός του πότε πρέπει να εκτελεστεί κάθε κόμβος στον αντίστοιχο επεξεργαστή.
- Γραμμικό χρονοδιάγραμμα.
- Υπολογίζεται με τη μέθοδο των υπερ-επιπέδων.
- Πρώτα βρίσκουμε τα διανύσματα εξάρτησης $\vec{\mathbf{d}}$ (βλέπε παράλληλα loops - διάλεξη 11).

Χρονοδιάγραμμα - Εξαρτήσεις

- Αλγόριθμος πολλαπλασιασμού πίνακα με διάνυσμα:
- `c[i][j] = c[i][j-1] + A[i][j]*b[j];`
- Έστω τα στιγμιότυπα: $\vec{\boldsymbol{i}} = [i1, i2]$, $\vec{\boldsymbol{j}} = [j1, j2]$ με το $\vec{\boldsymbol{i}}$ να προηγείται του $\vec{\boldsymbol{j}}$.

Εγγραφή Ανάγνωση

```
c[i1][i2] = c[i1][i2-1] + A[i1][i2]*b[i2];
c[j1][j2] = c[j1][j2-1] + A[j1][j2]*b[j2];
```

- Περίπτωση εξάρτησης: $[i1, i2] = [j1, j2 - 1]$
- $i1 = j1 \;\; \& \;\; i2 = j2 - 1 \Rightarrow i2 < j2$
- Εξάρτηση ροής, $\vec{\boldsymbol{d}} = \vec{\boldsymbol{j}} - \vec{\boldsymbol{i}} = [j1 - i1, j2 - i2] = [0 \;\; 1]$

Εσωτερικό γινόμενο - ιδιότητες

- **Ορισμός:** Εσωτερικό γινόμενο 2 διανυσμάτων:

  2 Διανύσματα: $\vec{\boldsymbol{a}} = [a_1, a_2], \vec{\boldsymbol{b}} = [b_1, b_2]$

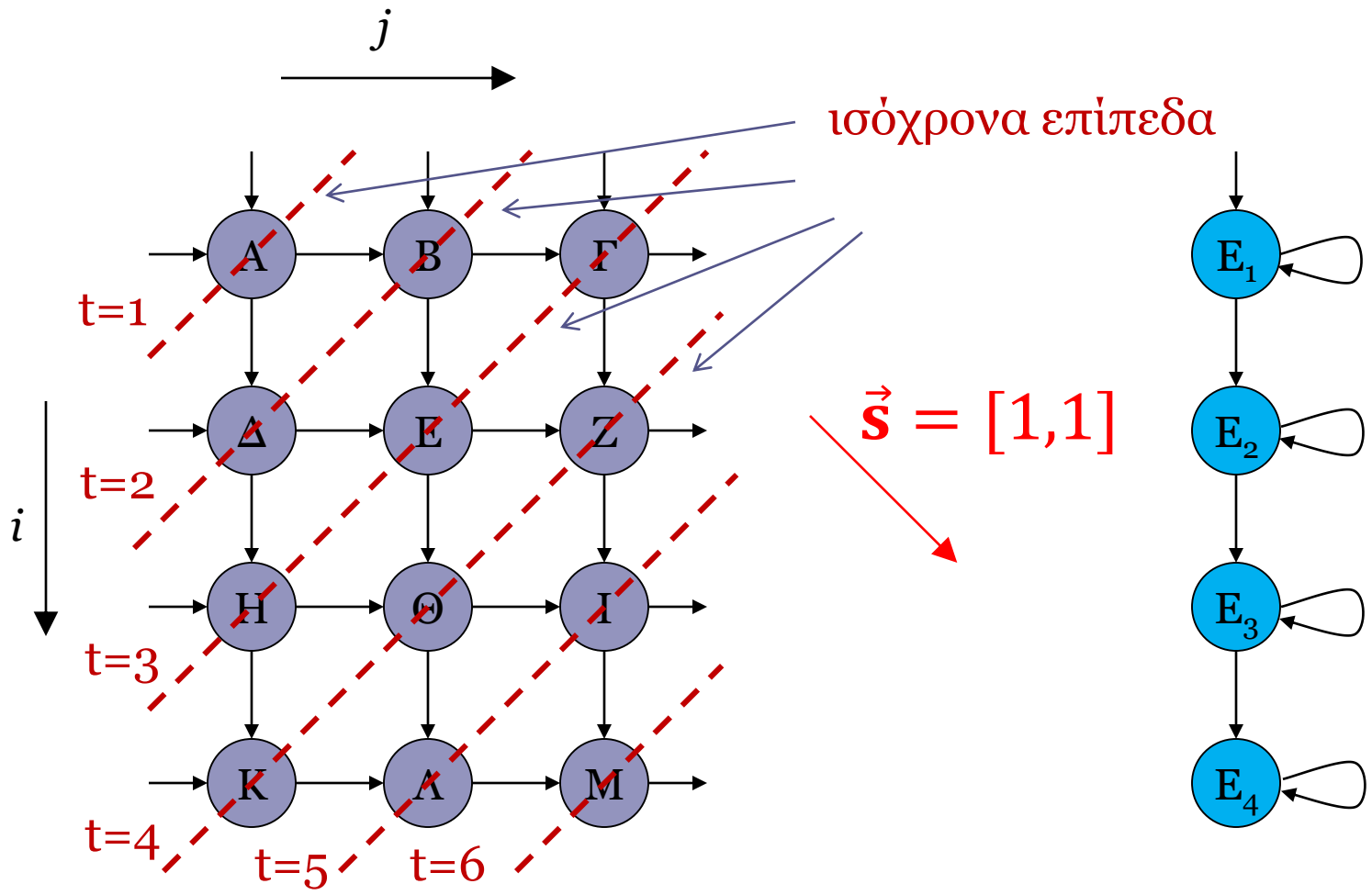
  Εσωτερικό γινόμενο: $\langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \rangle = \vec{\boldsymbol{a}} \cdot \vec{\boldsymbol{b}} = a_1 b_1 + a_2 b_2.$

- **Ιδιότητες:**
  - Δύο διανύσματα **a**, **b** είναι κάθετα $\Leftrightarrow \langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \rangle = \mathbf{0}$
  - Τα **a**, **b** σχηματίζουν γωνία $< 90° \Leftrightarrow \langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \rangle > \mathbf{0}$
  - Τα **a**, **b** σχηματίζουν γωνία $> 90° \Leftrightarrow \langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \rangle < \mathbf{0}$
  - Τα διανύσματα $\vec{\boldsymbol{a}} = [x, y], \vec{\boldsymbol{b}} = [y, -x]$ είναι κάθετα.

Διάνυσμα Χρονοδρομολόγησης

- Ορίζουμε ένα διάνυσμα $\vec{\mathbf{s}}$ ώστε $\vec{\mathbf{s}} \cdot \vec{\boldsymbol{d}} > 0$.
- Έστω $\vec{\mathbf{s}} = [1 \;\; 1]$.
- Ορίζονται ισόχρονα επίπεδα κάθετα στο $\vec{\mathbf{s}}$.

Χρονοδιάγραμμα - Ισόχρονα επίπεδα



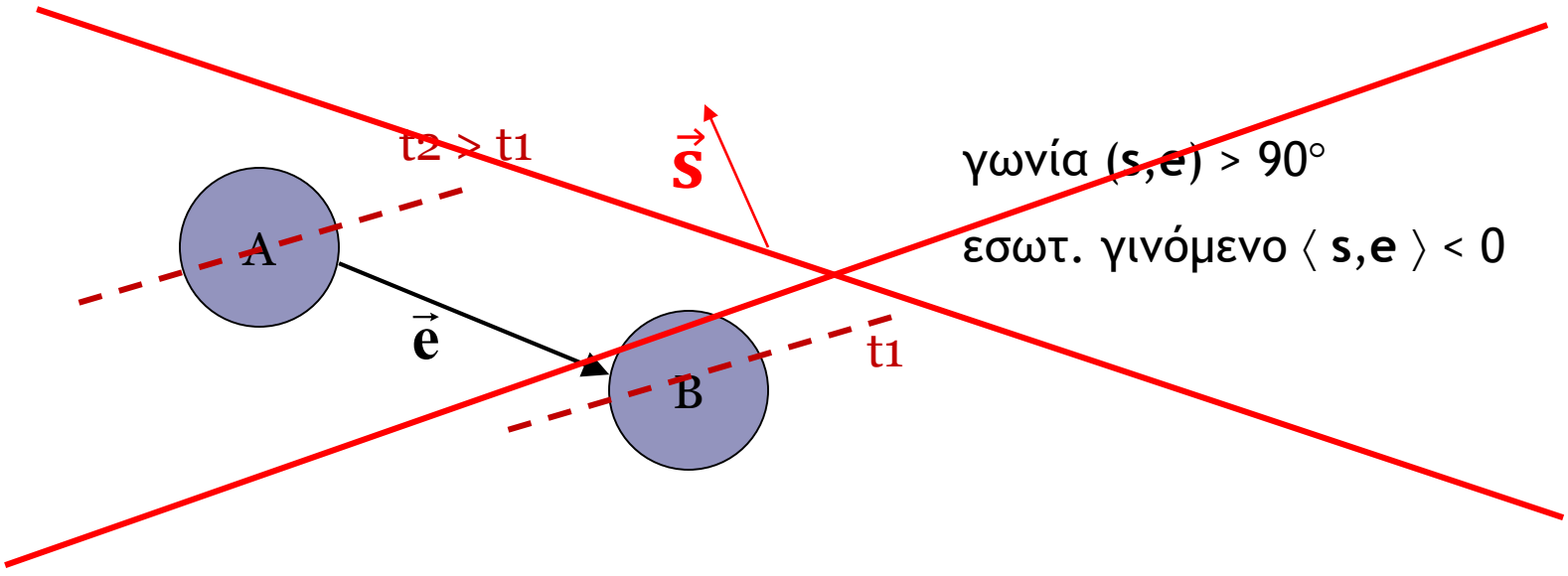
Ισόχρονα επίπεδα

- Ισόχρονο επίπεδο t = όλοι οι κόμβοι που βρίσκονται πάνω σ' αυτό εκτελούνται την ίδια χρονική στιγμή t.
- Όλα τα ισόχρονα επίπεδα είναι κάθετα στο χρονοδιάγραμμα $\vec{\mathbf{s}}$.
- Αν $\vec{\boldsymbol{t}}$ είναι διάνυσμα παράλληλο με τα ισόχρονα επίπεδα τότε:

$$\langle \vec{\mathbf{s}}, \vec{\boldsymbol{t}} \rangle = \mathbf{0}$$

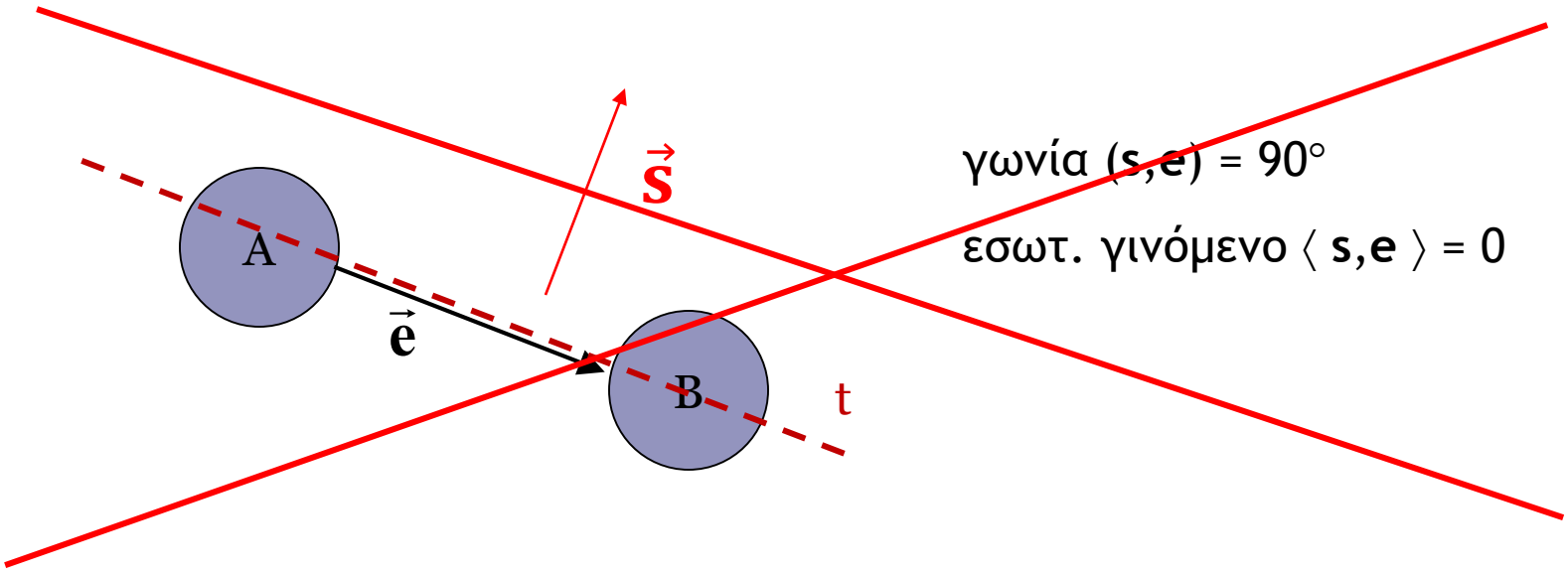
Περιορισμοί (1)

(α) Απαγορεύεται να εκτελέσουμε το Β πριν από το Α αν το Β εξαρτάται λογικά από το Α.



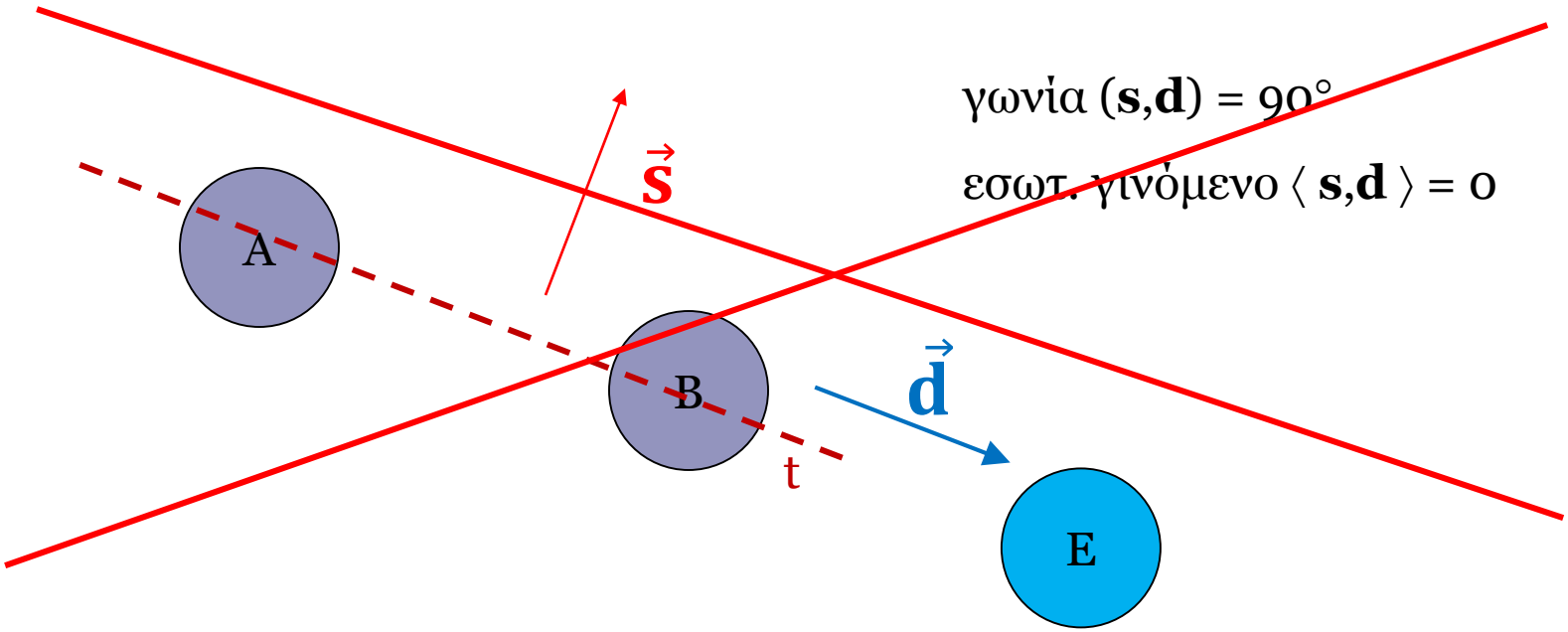
Περιορισμοί (2)

(β) Απαγορεύεται να εκτελέσουμε το Β ταυτόχρονα με το Α αν το Β εξαρτάται λογικά από το Α



Περιορισμοί (3)

(γ) Απαγορεύεται να εκτελέσουμε το Β ταυτόχρονα με το Α αν και το Β και το Α εκτελούνται στον ίδιο επεξεργαστή.



Περιορισμοί (4)

- (α) Για όλες τις ακμές $\vec{\mathbf{e}_i}$ πρέπει να ισχύει:

$$\langle \vec{\mathbf{s}}, \vec{\mathbf{e}_i} \rangle > \mathbf{0}$$

- (β) Για το διάνυσμα προβολής $\vec{\mathbf{p}}$ πρέπει να ισχύει:

$$\langle \vec{\mathbf{s}}, \vec{\mathbf{p}} \rangle \neq \mathbf{0}$$

- Σε διαφορετική περίπτωση το χρονοδιάγραμμα $\vec{\mathbf{s}}$ καλείται μη αποδεκτό.

Αποδεκτό χρονοδιάγραμμα

- Στο συγκεκριμένο παράδειγμα:

$$\vec{\mathbf{s}} = [1 \;\; 1], \vec{\boldsymbol{d}} = [0 \;\; 1]$$

- Ο γράφος εξάρτησης έχει δύο είδη ακμών:

$$\overrightarrow{\mathbf{e_1}} = [1 \;\; 0], \overrightarrow{\mathbf{e_2}} = [0 \;\; 1]$$

- Έχουμε:

$$\langle \vec{\mathbf{s}}, \overrightarrow{\mathbf{e_1}} \rangle = 1 \cdot 1 + 1 \cdot 0 = 1 > 0$$
$$\langle \vec{\mathbf{s}}, \overrightarrow{\mathbf{e_2}} \rangle = 0 \cdot 1 + 1 \cdot 1 = 1 > 0$$
$$\langle \vec{\mathbf{s}}, \vec{\mathbf{p}} \rangle = 1 \cdot 0 + 1 \cdot 1 = 1 \neq 0$$

s αποδεκτό

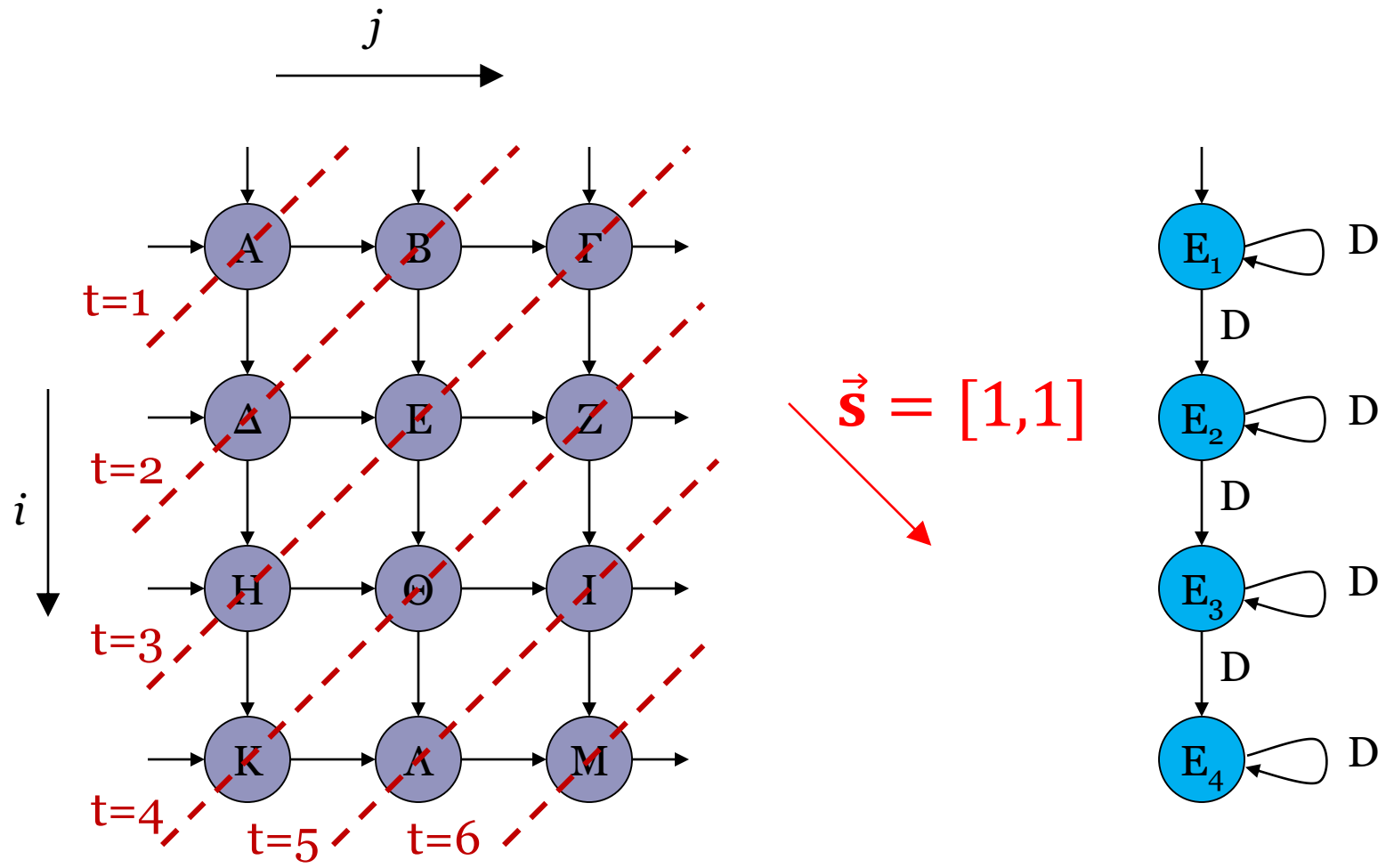
Πίνακας Χρονισμού

χρόνος →

επεξεργαστές ↓

	t=1	t=2	t=3	t=4	t=5	t=6
E1	Α	Β	Γ	-	-	-
E2	-	Δ	Ε	Ζ	-	-
E3	-	-	Η	Θ	Ι	-
E4	-	-	-	Κ	Λ	Μ

Προσθήκη delays στον SFG



Αλγόριθμος

- Αν μας δίνεται ο κώδικας μπορούμε να φτιάξουμε τον γράφο εξάρτησης.
- Αντίστροφα, από το γράφο εξάρτησης μπορούμε να ανακατασκευάσουμε τον κώδικα.
- Αν μας δίνεται ο γράφος εξάρτησης και το διάνυσμα προβολής τότε μπορούμε να φτιάξουμε το γράφο ροής σήματος.
- Αντίστροφα, αν μας δίνεται γράφος ροής σήματος, το διάνυσμα προβολής και το χρονοδιάγραμμα μπορούμε να φτιάξουμε το γράφο εξάρτησης.
- Καμία απώλεια πληροφορίας.

Σχεδίαση Συστολικού Επεξεργαστή