Cuprins

5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

Tipuri și nivele de paralelism (1)

- Soluțiile unor probleme pot conține două tipuri de paralelism:
  - Paralelism funcțional: derivă din logica unei soluții a problemei; apare în toate descrierile formale ale soluției
  - Paralelism de date: derivă din utilizarea structurilor de date care permit operații în paralel asupra elementelor de date
    - Este inerent numai într-un set restrâns de probleme; exemplu: prelucrarea imaginilor

Tipuri și nivele de paralelism (2)

- Paralelismul funcțional se poate utiliza la patru nivele de granularitate
- Paralelism la nivelul instrucțiunilor
  - Execuția instrucțiunilor în paralel
  - Granularitate fină
  - Arhitecturi ILP (*Instruction-Level Parallelism*)
  - Înainte de execuție, acest paralelism trebuie detectat fie de către un compilator dedicat, fie de către arhitectură

Tipuri și nivele de paralelism (3)

- Paralelism la nivelul firelor de execuție
  - Fir de execuție (*thread*): fragment de cod; se poate executa în paralel cu alte fragmente
  - Firele de execuție sunt create în cadrul proceselor (programelor); partajează resursele procesului în care sunt create
  - Pot fi create de programator sau SO, sau pot fi generate de un compilator paralel
  - Pot fi executate în paralel de arhitecturi cu fire de execuție multiple

Tipuri și nivele de paralelism (4)

- Paralelism la nivelul proceselor
  - Proces: fragment de cod; are o granularitate mai mare comparativ cu un fir de execuție
  - Procesele pot fi create și executate în mod similar cu firele de execuție
- Paralelism la nivelul utilizatorilor
  - Utilizatorii multipli sunt independenți unii față de alții
  - Granularitate mare

5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

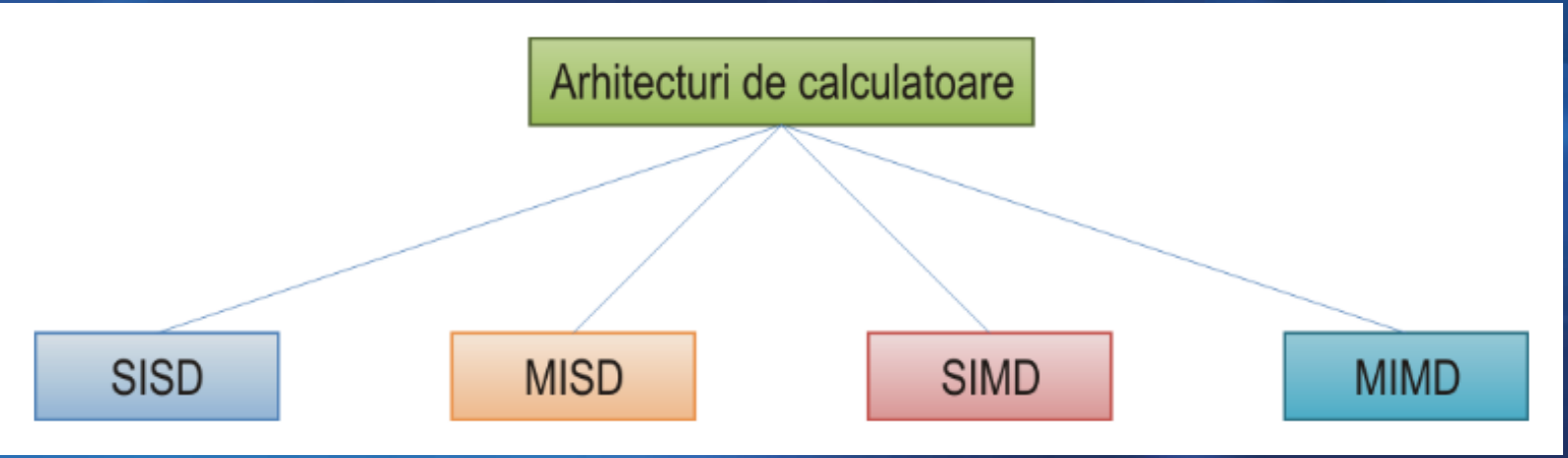
Clasificarea arhitecturilor paralele

- Clasificarea arhitecturilor paralele
  - Taxonomia Flynn
  - Taxonomie extinsă

Taxonomia Flynn (1)

- Prof. Michael Flynn a clasificat arhitecturile de calculatoare în patru categorii
- Criteriul: prezența unui singur șir sau a mai multor șiruri de instrucțiuni și de date
- Șir de instrucțiuni: set de instrucțiuni secvențiale executate de un procesor
- Șir de date: fluxul secvențial de date necesar șirului de instrucțiuni

Taxonomia Flynn (2)



Taxonomia Flynn (3)

- SISD (*Single Instruction stream, Single Data stream*)
  - Corespunde arhitecturii *von Neumann*
  - Se execută în orice moment o singură instrucțiune → calculatoare seriale scalare
  - Utilizează un registru *contor de program*
  - Actualmente există doar un număr redus de calculatoare SISD → se utilizează paralelismul în scopul creșterii eficienței

Taxonomia Flynn (4)

- MISD (*Multiple Instruction stream, Single Data stream*)
  - Mai multe instrucțiuni operează asupra unei singure date
  - Două posibilități de interpretare a structurii calculatoarelor MISD
  - Prima posibilitate: mai multe unități de prelucrare primesc instrucțiuni distincte care operează asupra acelorași date → clasă nepractică

Taxonomia Flynn (5)

- A doua posibilitate: o clasă de calculatoare în care șirul de date este trecut printr-o serie de unități de prelucrare
- Exemple: arhitecturi vectoriale sau arhitecturi *pipeline*
  - Execută o prelucrare vectorială utilizând o serie de etaje
  - Fiecare din acestea execută o anumită funcție și produce un rezultat intermediar

Taxonomia Flynn (6)

- SIMD (*Single Instruction stream, Multiple Data stream*)
  - O singură instrucțiune prelucrează simultan date diferite
  - Există mai multe procesoare de prelucrare și un singur procesor de control
  - Exemplu: procesoare matriceale
  - Calculatoarele SIMD pot executa și ele prelucrări vectoriale

Taxonomia Flynn (7)

- MIMD (*Multiple Instruction stream, Multiple Data stream*)
  - Arhitecturi cu mai multe procesoare de prelucrare
  - Mai multe instrucțiuni pot opera simultan asupra unor date diferite
  - Arhitecturile MIMD obțin o eficiență ridicată prin prelucrare paralelă
    - Procesoare multiple și procese multiple

Taxonomia Flynn (8)

- Taxonomia Flynn s-a dovedit utilă pentru clasificarea arhitecturilor de calculatoare
  - Totuși, nu ține cont de tipul de paralelism utilizat și de nivelul de granularitate
  - Au apărut arhitecturi care nu pot fi clasificate în mod clar prin această taxonomie
  - Nu clasifică în mod adecvat arhitecturile hibride și arhitecturile neconvenționale
- Au fost propuse mai multe taxonomii

Clasificarea arhitecturilor paralele

- Clasificarea arhitecturilor paralele
  - Taxonomia Flynn
  - Taxonomie extinsă

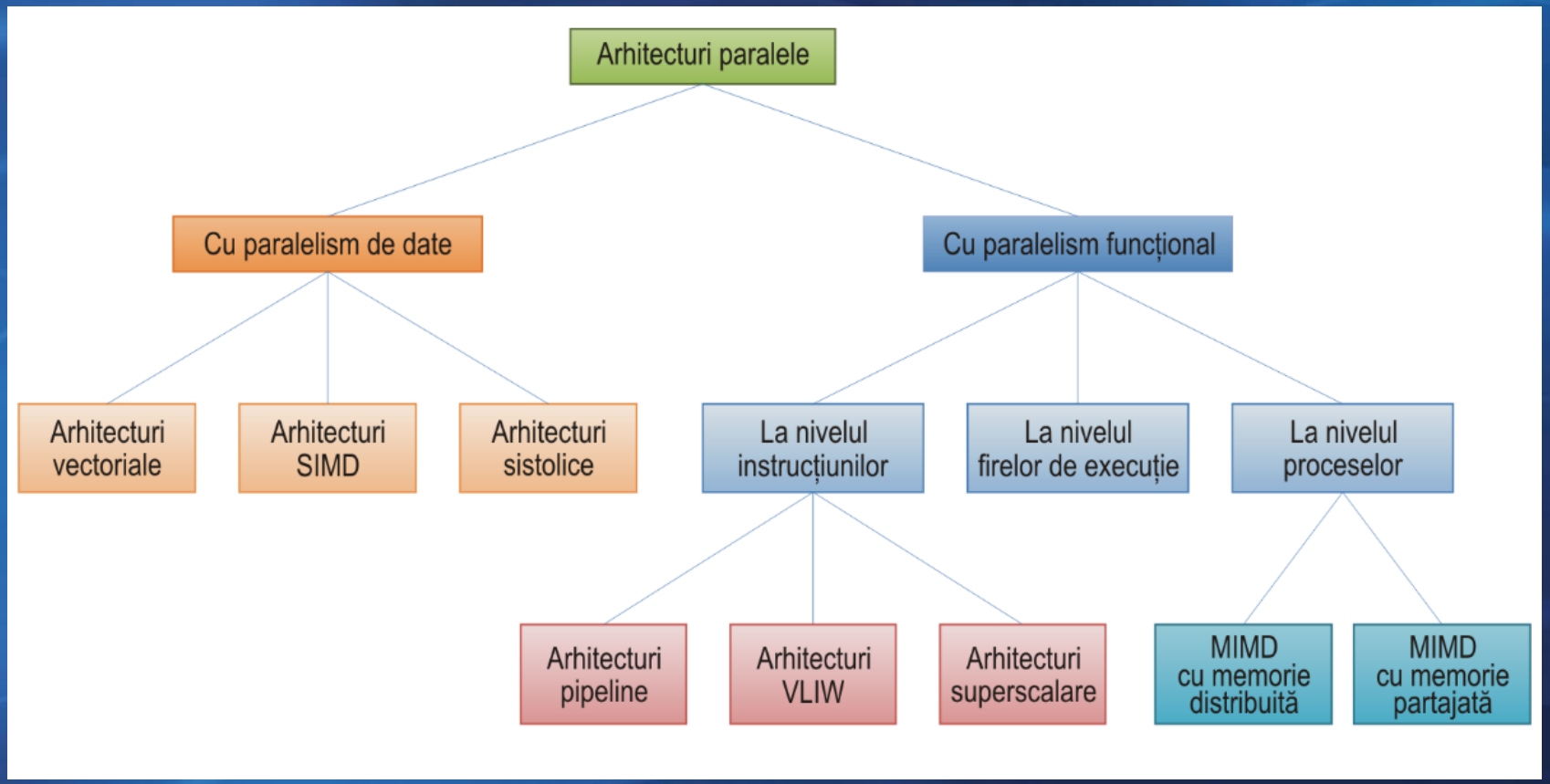
Taxonomie extinsă (1)

- Taxonomia prezentată se bazează pe tipul de paralelism: funcțional sau de date
  - Ține cont de granularitatea paralelismului funcțional utilizat
- Arhitecturi cu paralelism de date
  - Arhitecturi vectoriale
  - Arhitecturi SIMD
  - Arhitecturi sistolice

Taxonomie extinsă (2)

- Arhitecturi cu paralelism funcțional
  - Cu paralelism la nivelul instrucțiunilor: *pipeline*; VLIW (*Very Long Instruction Word*); superscalare
  - Cu paralelism la nivelul firelor de execuție: arhitecturi cu fire de execuție multiple → arhitecturi cu flux de date
  - Cu paralelism la nivelul proceselor: arhitecturi MIMD → cu memorie distribuită; cu memorie partajată

Taxonomie extinsă (3)



5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

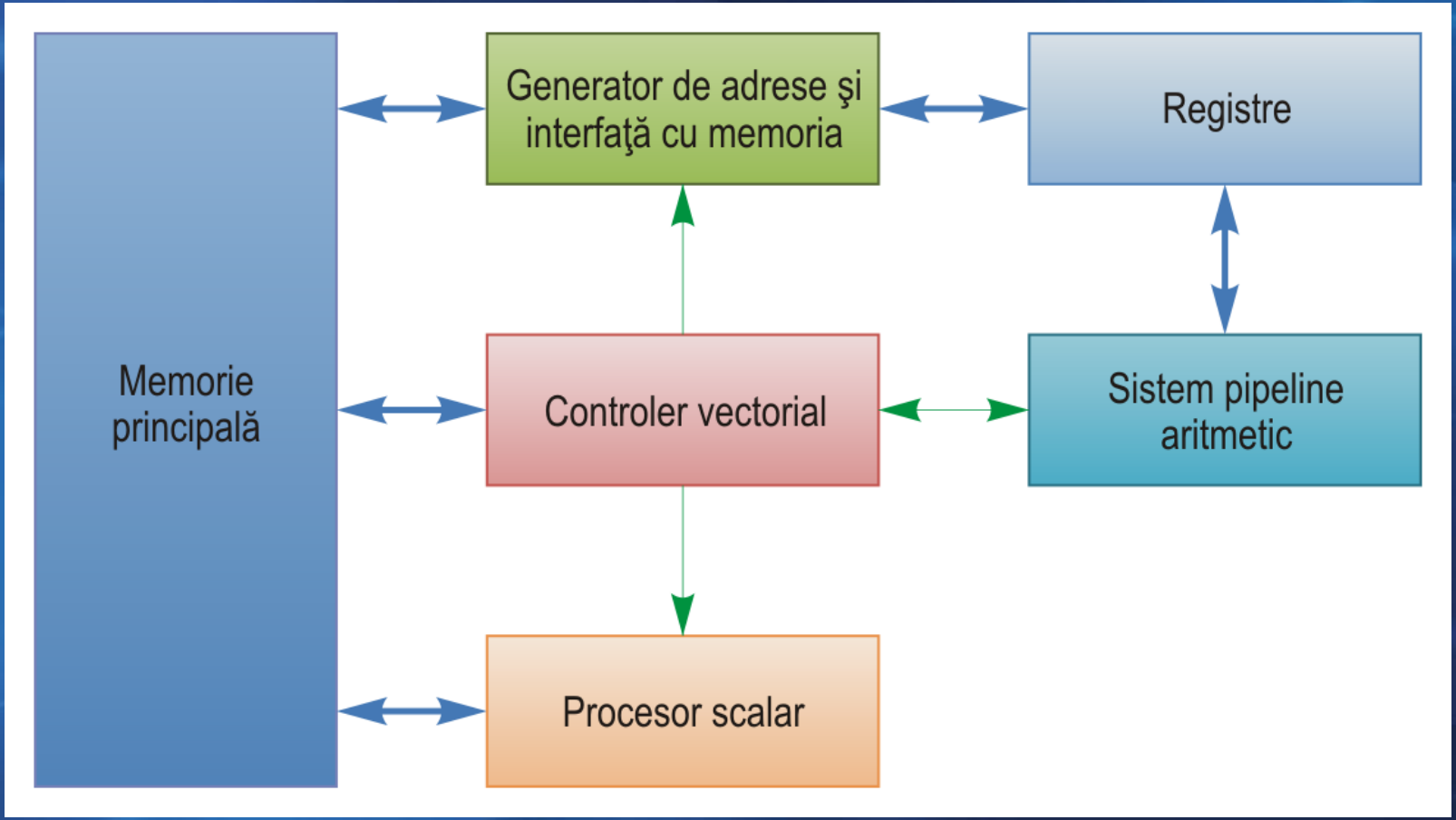
Arhitecturi vectoriale (1)

- Aparțin categoriei MISD
- Supercalculatoare optimizate pentru execuția operațiilor cu vectori sau tablouri
  - Prelucrarea unuia sau a mai multor vectori pentru a obține un rezultat scalar
  - Combinarea a doi vectori
  - Combinarea unui scalar cu un vector, rezultând un nou vector
  - O combinație a operațiilor anterioare

Arhitecturi vectoriale (2)

- Unitatea centrală utilizează în mod extensiv tehnica *pipeline*
- Operațiile asupra vectorilor se execută serial asupra fiecărui element
- Tehnica *pipeline* se utilizează și pentru accesul la memorie
- Setul de registre poate păstra toate elementele unuia sau a mai multor vectori

Arhitecturi vectoriale (3)



Arhitecturi vectoriale (4)

- Memoria principală
  - Organizarea memoriei este un factor esențial pentru performanța sistemului
  - Se utilizează memorii cu unități multiple și intercalarea adreselor de memorie
- Procesorul scalar
  - Execută toate operațiile scalare: controlul programului, funcțiile SO, operațiile de I/E
  - Comandă controlerul vectorial

Arhitecturi vectoriale (5)

- Controlerul vectorial
  - Decodifică instrucțiunile vectoriale
  - Inițiază operațiile generatorului de adrese și ale sistemului *pipeline* aritmetic
- Generatorul de adrese și interfața cu memoria
  - Asigură transferul rapid al datelor între sistemul *pipeline* aritmetic și memorie
  - Primește adresele logice ale vectorilor de la controlerul vectorial

Arhitecturi vectoriale (6)

- Registrele
  - Au rolul unor buffere între memoria principală și sistemul *pipeline* aritmetic
  - Permit un acces mai rapid la date
  - Operanzii vectoriali sunt transferați între memoria principală și registre în blocuri
- Sistemul *pipeline* aritmetic
  - Recepționează operanzii din registre și returnează rezultatele în registre

Arhitecturi vectoriale (7)

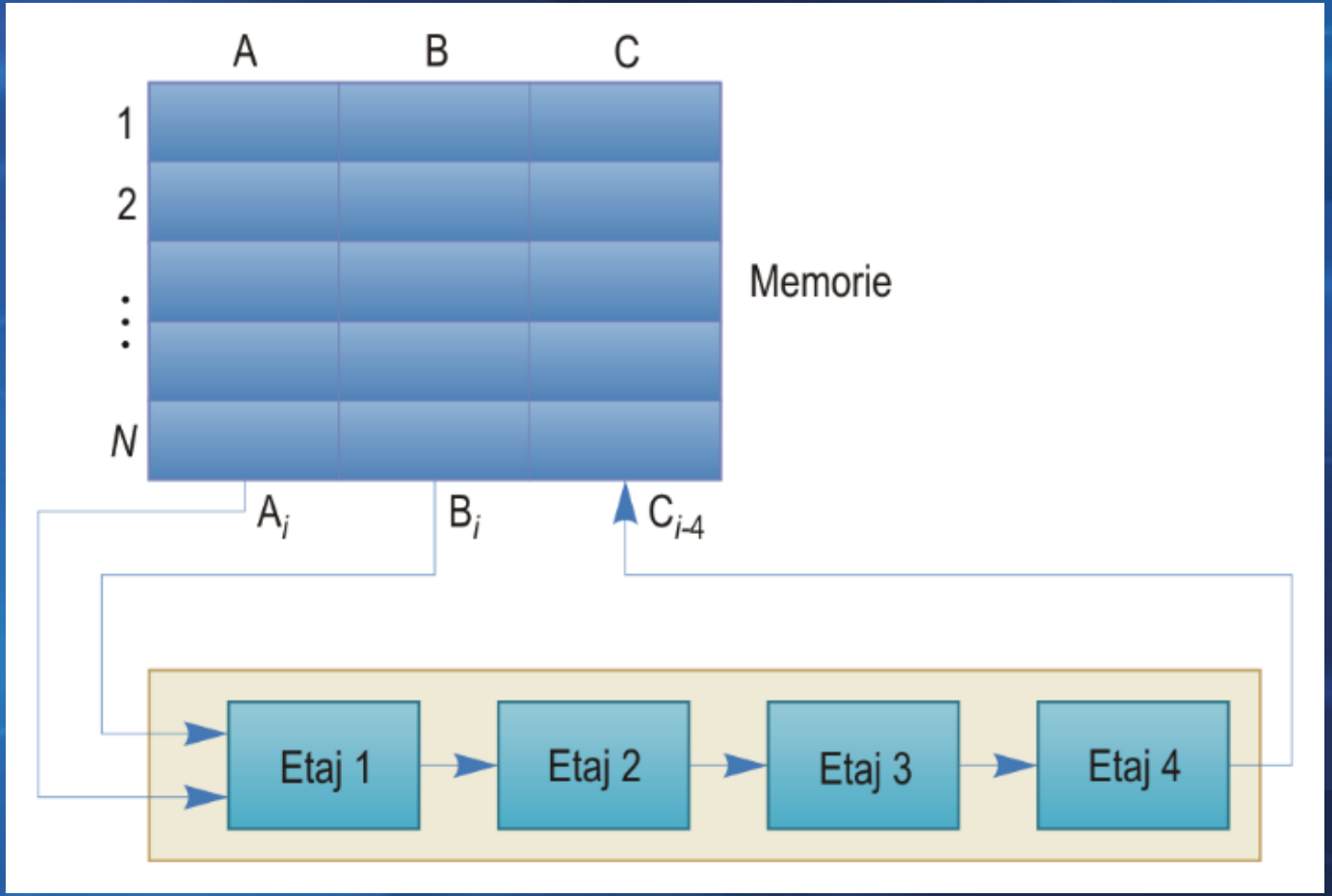
- Exemplu: Adunarea a doi vectori
  - **A**, **B**: vectori de câte *N* elemente
  - Vectorul sumă **C**: $C_i = A_i + B_i$ ($i = 1 .. N$)
  - Secvența pentru o arhitectură SISD:

```
for (i=1; i<=N; i++) {
    C[i] = A[i] + B[i];
}
```

  - Instrucțiunea vectorială echivalentă:

```
C = A + B;
```

Arhitecturi vectoriale (8)



Arhitecturi vectoriale (9)

- Caracteristici ale arhitecturilor vectoriale
  - Memorie de viteză ridicată
  - Număr mare de registre: vectoriale, scalare
  - Set de instrucțiuni conținând instrucțiuni vectoriale
  - Operații executate în paralel în diferite module:
    - Mai multe sisteme *pipeline* aritmetice
    - Procesorul scalar și modulul vectorial

Arhitecturi vectoriale (10)

- Controlerul vectorial și generatorul de adrese din modulul vectorial
- Interfețele între memorie și registre, respectiv între registre și sistemul *pipeline* aritmetic

- Exemple de arhitecturi vectoriale
  - Supercalculatoare ale firmelor Cray: Cray X-MP, Cray Y-MP, Cray X1
  - Seria SX de supercalculatoare ale firmei NEC Corporation: SX-ACE, SX-Aurora TSUBASA

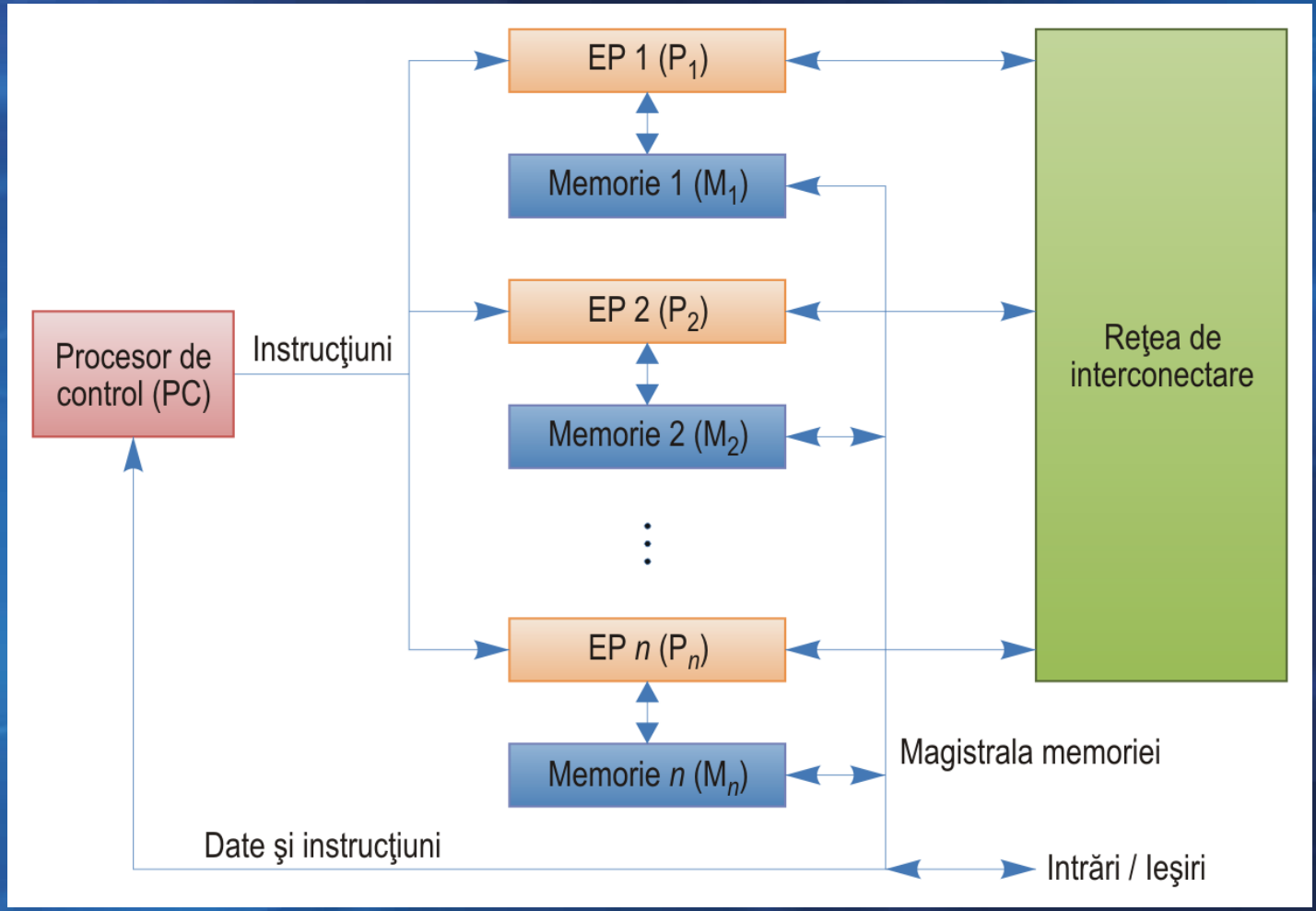
5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

Arhitecturi SIMD (1)

- Se numesc și procesoare matriceale
- Un singur procesor de control (PC) și elemente de procesare (EP) multiple
- În orice moment, fiecare EP execută aceeași instrucțiune $\rightarrow$ sincronizare
- Fiecare EP prelucrează propriul șir de date $\rightarrow$ paralelism de date
- EP sunt interconectate printr-o rețea de interconectare (RI)

Arhitecturi SIMD (2)



Arhitecturi SIMD (3)

- Memoria
  - Programele sunt distribuite în mai multe blocuri de memorie → intercalarea adreselor
  - Datele sunt și ele distribuite → fiecare EP are o memorie locală
  - Este necesar accesul simultan la memorie de către PC și diferitele EP → pot apare conflicte
  - Organizarea datelor în memorie este importantă pentru arhitecturile SIMD

Arhitecturi SIMD (4)

- O posibilitate: datele necesare pentru fiecare EP sunt disponibile în memoria sa locală
- Poate fi necesară rearanjarea datelor în memorie pe parcursul calculelor
- Exemplu: Însumarea elementelor din fiecare coloană a unei matrice
  - Fiecare EP însumează elementele dintr-o coloană $\rightarrow$ acces la elementele unei coloane
  - Dacă va fi necesar accesul la elementele unei linii, matricele trebuie rearanjate în memorie

Arhitecturi SIMD (5)

- Altă posibilitate: utilizarea unei rețele de interconectare între memorii și EP
  - Comutator: fiecare memorie de date poate fi accesată de oricare EP $\rightarrow$ alinierea datelor
- Procesorul de control
  - Încarcă instrucțiunile și le decodifică
  - Transferă instrucțiunile aritmetice și logice la EP pentru execuție
  - Generează semnale de control pentru EP
  - Execută instrucțiunile de control

Arhitecturi SIMD (6)

- Elementele de procesare (EP)
  - Execută operațiile aritmetice și logice
  - Sunt sincronizate la nivel de instrucțiune
  - Fiecare EP corespunde cu calea de date și UAL dintr-un procesor SISD
  - Unele EP pot fi dezactivate în timpul execuției anumitor instrucțiuni
  - Activarea și dezactivarea EP: de către PC sau logica locală din fiecare EP

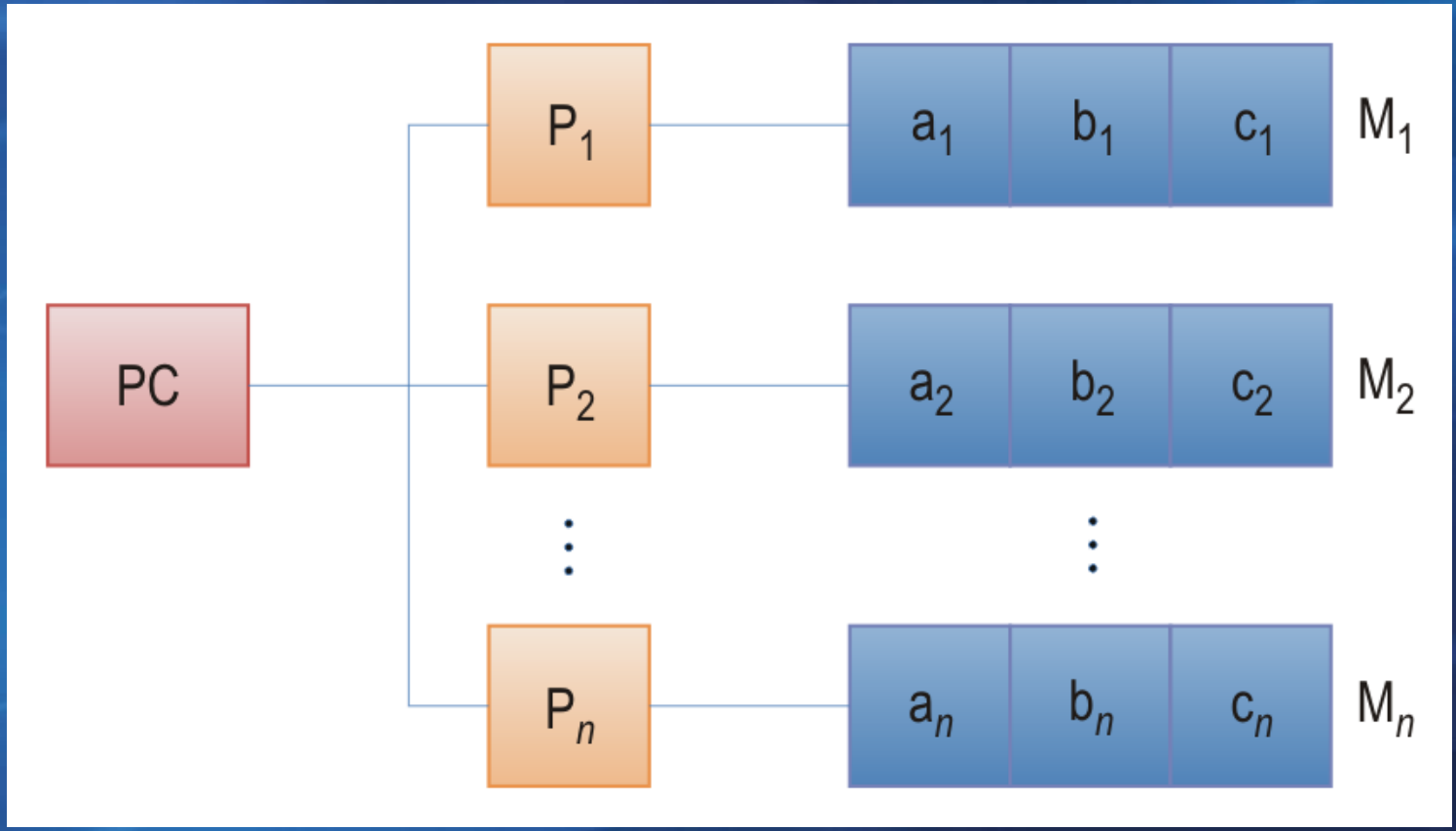
Arhitecturi SIMD (7)

- Rețeaua de interconectare (RI)
  - Trei tipuri de modele SIMD în funcție de RI
  - Comutator 1 : *n* între PC și cele *n* EP
  - Comutator 1 : *n* între PC și blocurile de mem.
  - Tipul 1 și Tipul 2
    - Fiecare EP are acces direct la memoria sa locală
    - Comutator *n* x *n* între toate EP
  - Tipul 3
    - Nu există interconexiune directă între EP
    - Comutator *n* x *n* între EP și blocurile de memorie

Arhitecturi SIMD (8)

- Exemplu: Adunarea a doi vectori
  - **A**, **B**: vectori de câte *N* elemente
  - Vectorul sumă **C**: $C_i = A_i + B_i$ ($i = 1 .. N$)
  - Se poate implementa printr-o arhitectură SIMD cu *N* elemente de procesare EP
  - RI nu este necesară
  - Elementele vectorilor **A** și **B** trebuie distribuite în *N* blocuri de memorie
  - Instrucțiunea SIMD: `C = A + B;`

Arhitecturi SIMD (9)



Arhitecturi SIMD (10)

- Primele arhitecturi SIMD: supercalculatoare
  - Calculatoarele Connection Machine ale firmei Thinking Machines Corp. (CM-1, CM-2)
  - Un număr mare de procesoare simple (CM-2: 65.536 de procesoare)
- Procesoarele actuale conțin seturi de instrucțiuni SIMD
  - Permit operații cu vectori de lungime relativ redusă: 16 .. 64 elemente

Arhitecturi SIMD (11)

- Exemple de seturi de instrucțiuni SIMD
  - VIS (*Visual Instruction Set*): Sun Microsystems pentru procesoarele UltraSPARC
  - MMX: Intel pentru arhitectura x86
  - AltiVec: Apple, IBM, Motorola pentru procesoarele PowerPC, POWER
  - 3DNow!: AMD pentru arhitectura x86
  - SSE (*Streaming SIMD Extensions*): Intel pentru arhitectura x86; versiuni ulterioare: SSE2, SSE3, SSE4

5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

Arhitecturi sistolice (1)

- Sunt formate prin interconectarea unui set de celule identice de prelucrare a datelor
  - Cuvintele de date circulă în mod sincron de la celulă la celulă
  - Fiecare celulă execută o etapă de prelucrare
  - Datele sunt complet prelucrate atunci când rezultatele apar de la celulele marginale
- O arhitectură sistolică unidimensională: similară cu un sistem *pipeline* cu etaje identice

Arhitecturi sistolice (2)

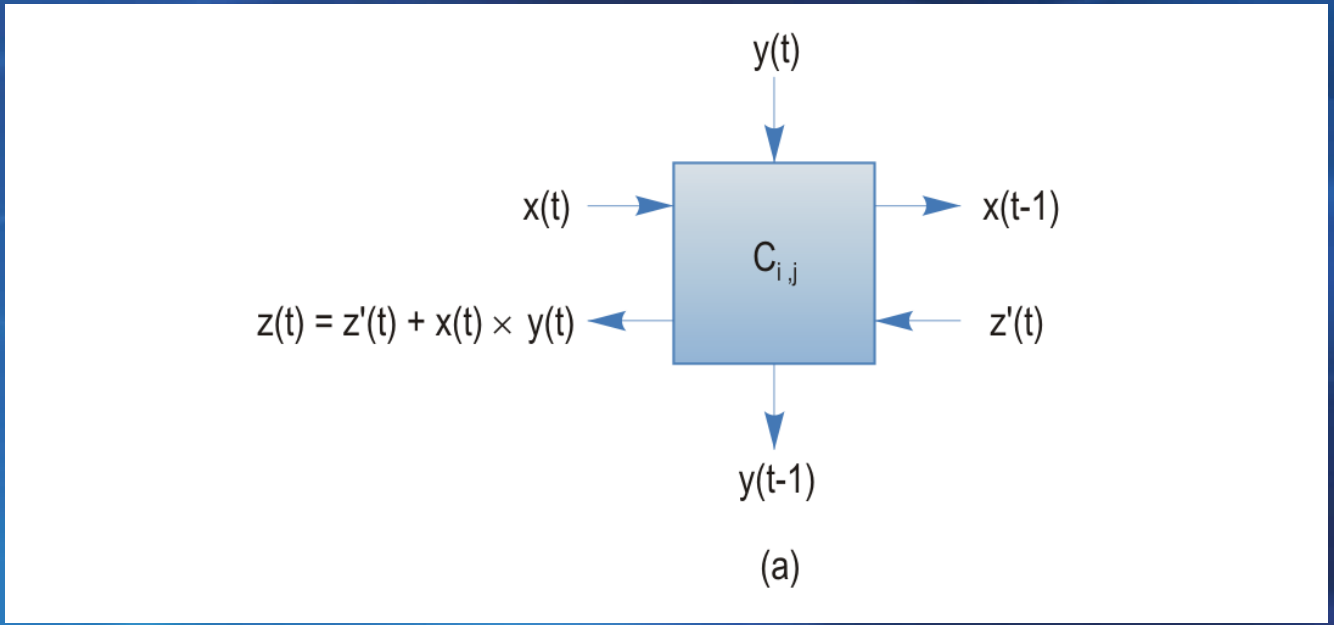
- Datele pot circula între celule în mai multe direcții în același timp
- Trebuie incluse buffere între celule
- Procesoare sistolice pentru implementarea unor operații aritmetice complexe:
  - Înmulțirea matricelor
  - Rezolvarea ecuațiilor liniare
  - Convoluția

Arhitecturi sistolice (3)

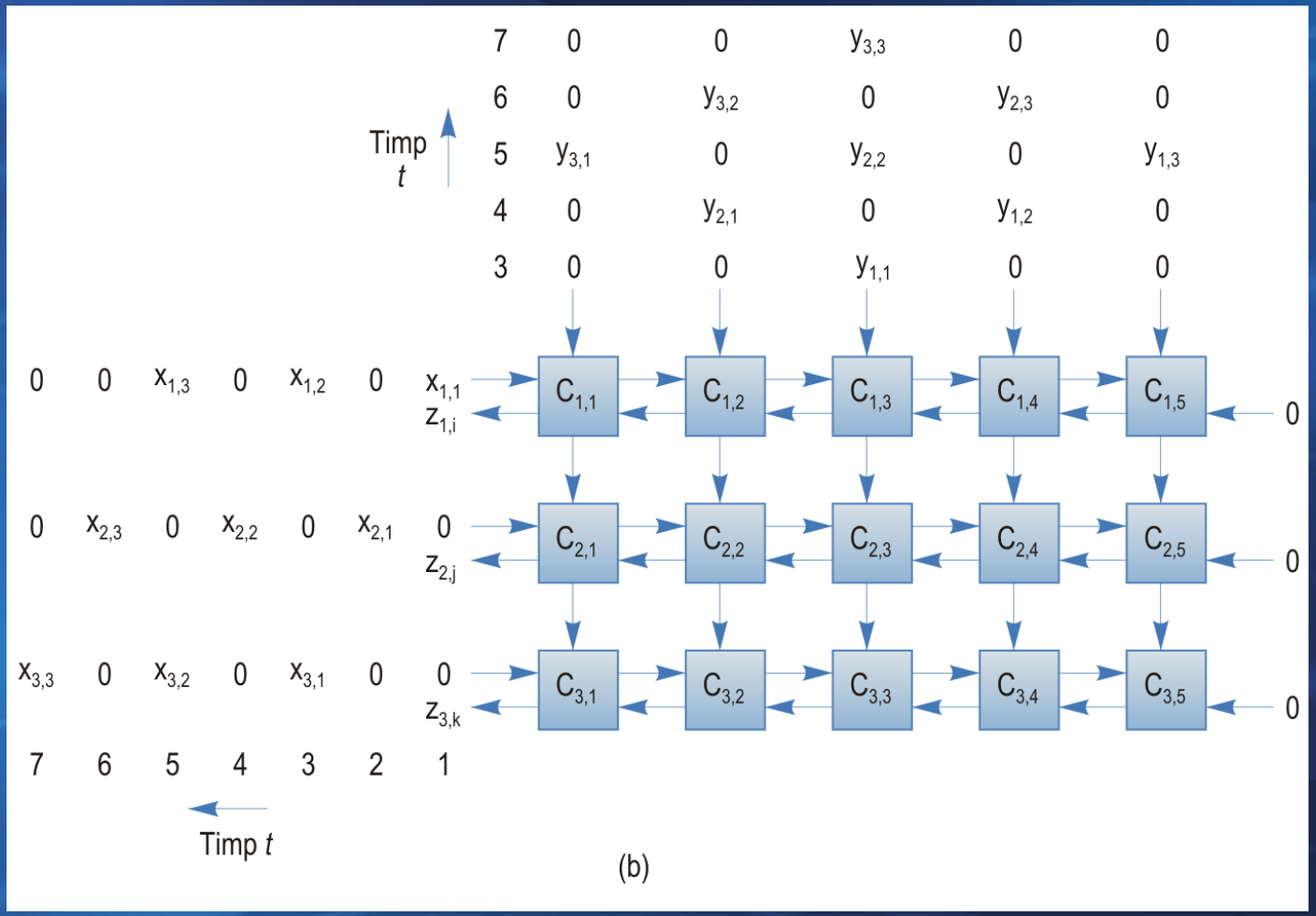
- Exemplu: Arhitectură bidimensională pentru înmulțirea a două matrice
  - $\boldsymbol{X} = [x_{i,j}]$ o matrice $n \times n$ de numere întregi sau în VM
  - Produsul matricei $\boldsymbol{X}$ cu matricea $\boldsymbol{Y} = [y_{i,j}]$ este matricea $\boldsymbol{Z} = [z_{i,j}]$:

    $$z_{i,j} = \sum_{k=1}^{n} x_{i,k} \times y_{k,j}$$
  - Celula de bază execută operația de înmulțire și adunare: $z := z' + x \times y$

Arhitecturi sistolice (4)



Arhitecturi sistolice (5)



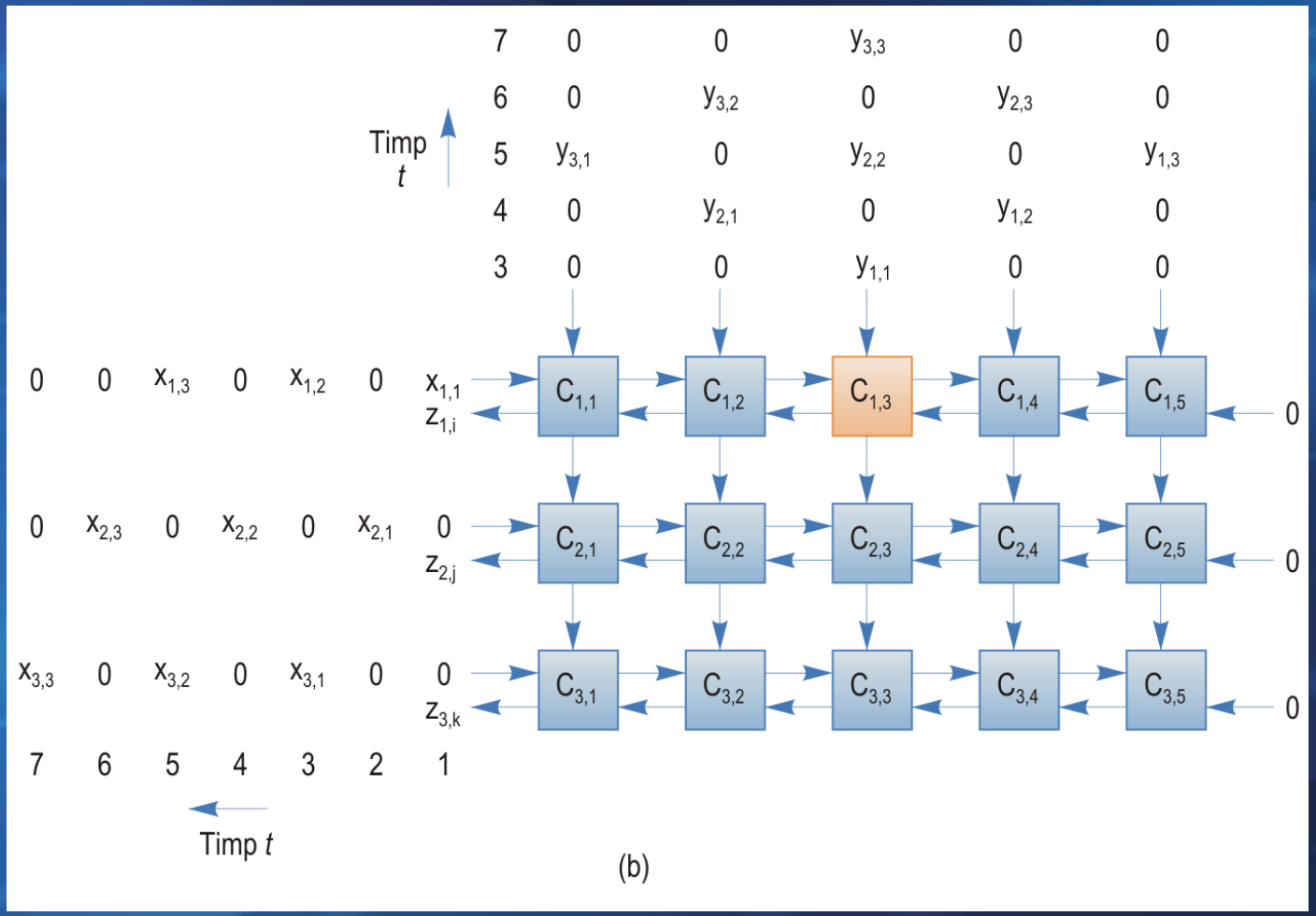
Arhitecturi sistolice (6)

- Calculul termenului $z_{1,1}$:

  $$z_{1,1} = x_{1,1} \times y_{1,1} + x_{1,2} \times y_{2,1} + x_{1,3} \times y_{3,1}$$

- $t = 1..2$
  - Operandul $x_{1,1}$ circulă spre dreapta, întâlnind numai valori zero ale y și z
- $t = 3$
  - Operandul $x_{1,1}$ întâlnește $y_{1,1}$ la celula $C_{1,3}$
  - Aceasta calculează valoarea $z = 0 + x_{1,1} \times y_{1,1}$ pe care o trimite la celula $C_{1,2}$
  - Celula $C_{1,3}$ expediază $y_{1,1}$ la $C_{2,3}$ și $x_{1,1}$ la $C_{1,4}$

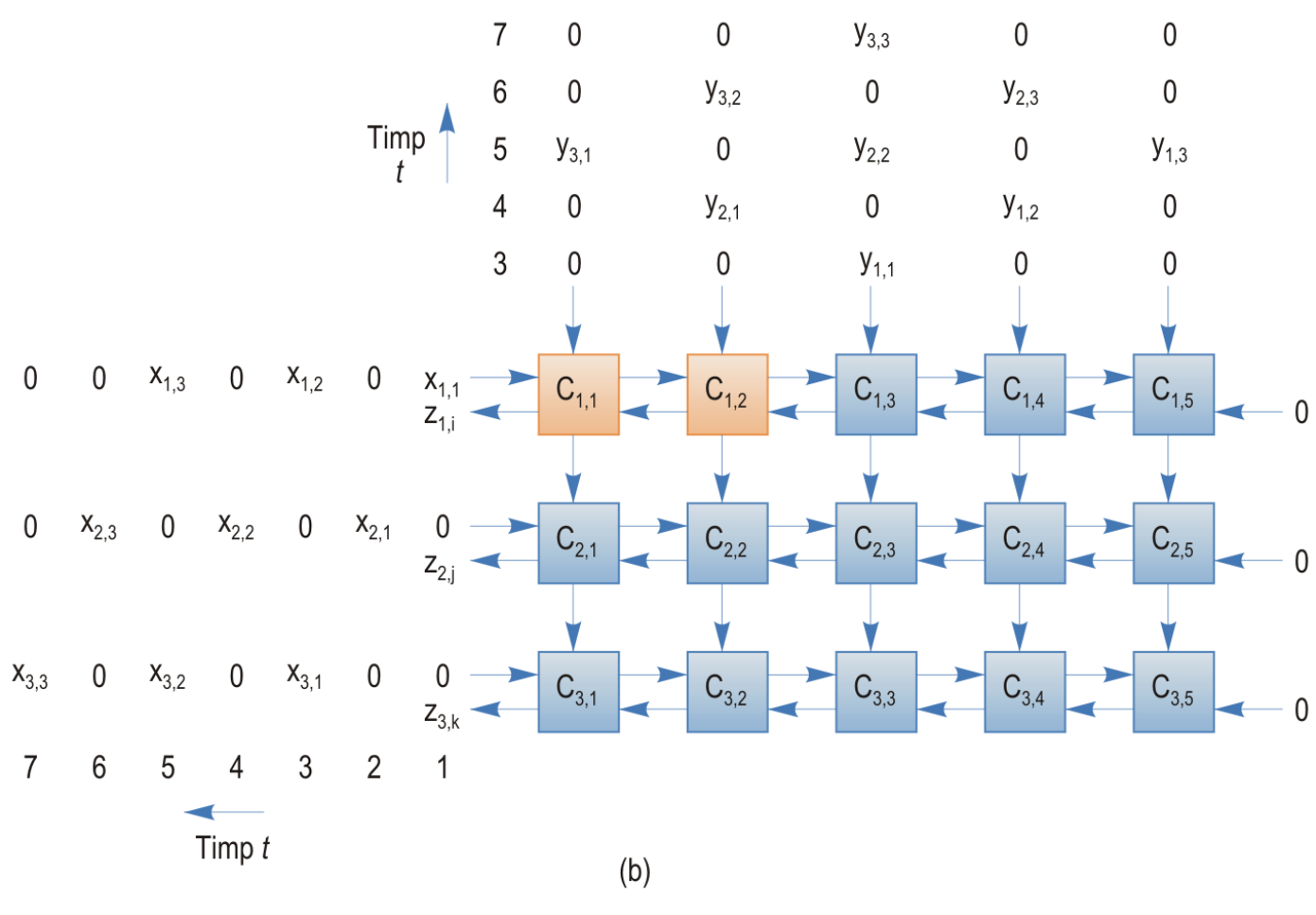
Arhitecturi sistolice (7)



Arhitecturi sistolice (8)

- $t = 4$
  - Operanzii $x_{1,2}$ și $y_{2,1}$ se aplică la intrările $C_{1,2}$
  - Această celulă calculează valoarea $z = z' + x_{1,2} \times y_{2,1}$, unde $z' = x_{1,1} \times y_{1,1}$
- $t = 5$
  - Ultima pereche de operanzi $x_{1,3}$ și $y_{3,1}$ converge la celula marginală $C_{1,1}$
  - Aceasta calculează valoarea $z = z' + x_{1,3} \times y_{3,1}$, utilizând valoarea furnizată de $C_{1,2} \Rightarrow z_{1,1}$

Arhitecturi sistolice (9)



Arhitecturi sistolice (10)

- $t = 6$
  - $C_{1,1}$ furnizează valoarea zero
- $t = 7$
  - $C_{1,1}$ furnizează următorul element $z_{1,2}$
- Procesul continuă până când se generează toate elementele primei linii a matricei ***Z***
- În paralel, celelalte rânduri de celule calculează liniile rămase ale matricei ***Z***

Arhitecturi sistolice (11)

- Caracteristici ale arhitecturilor sistolice:
  - Asigură un grad ridicat de paralelism
  - Seturile de date prelucrate parțial circulă în mod sincron prin arhitectura sistolică, eventual în mai multe direcții
  - Se simplifică implementarea într-o singură capsulă $\rightarrow$ celule și interconexiuni uniforme
  - Controlul arhitecturii este simplu
  - Datele trebuie furnizate în secvența corectă

Arhitecturi sistolice (12)

- Dacă matricele ***X*** și ***Y*** sunt generate în timp real, nu este necesară memorarea lor înainte de a calcula ***X*** × ***Y***
- Necesarul de circuite pentru implementare este relativ ridicat

- Aplicații:
  - Proiectarea circuitelor aritmetice speciale pentru prelucrarea digitală a semnalelor → prelucrare în timp real

Rezumat (1)

- Există două tipuri de paralelism care se pot utiliza: paralelism funcțional și paralelism de date
- Paralelismul funcțional se poate utiliza la diferite nivele de granularitate: instrucțiuni, fire de execuție, procese, utilizatori
- Taxonomia Flynn clasifică arhitecturile de calculatoare în funcție de numărul șirurilor de instrucțiuni și de date
  - Nu ține cont de tipul de paralelism utilizat

Rezumat (2)

- Taxonomia extinsă realizează distincția între arhitecturi cu paralelism funcțional și cele cu paralelism de date
- Arhitecturile vectoriale sunt optimizate pentru execuția operațiilor cu vectori sau tablouri
  - Operațiile asupra vectorilor se execută serial
- Arhitecturile SIMD conțin EP multiple care sunt sincronizate la nivel de instrucțiune
  - Organizarea datelor în memorie influențează semnificativ performanța

Rezumat (3)

- Elementele unor vectori sau tablouri sunt prelucrate simultan
- Procesoarele actuale conțin instrucțiuni SIMD
- Arhitecturile sistolice sunt avantajoase pentru prelucrarea digitală a semnalelor
  - Conțin un număr mare de celule de prelucrare interconectate într-un mod simplu
  - Datele prelucrate parțial circulă între celule în mod sincron, în mai multe direcții
  - Rezultatele finale apar la marginile matricei de celule

Noțiuni, cunoștințe (1)

- Paralelism funcțional și paralelism de date
- Nivele de granularitate ale paralelismului funcțional
- Taxonomia Flynn
- Taxonomie extinsă
- Principiul arhitecturilor vectoriale
- Schema bloc a unei arhitecturi vectoriale
- Componentele unei arhitecturi vectoriale
- Caracteristici ale arhitecturilor vectoriale

Noțiuni, cunoștințe (2)

- Principiul arhitecturilor SIMD
- Schema bloc a unei arhitecturi SIMD
- Componentele unei arhitecturi SIMD
- Exemple de seturi de instrucțiuni SIMD
- Principiul arhitecturilor sistolice
- Exemplu de arhitectură sistolică pentru înmulțirea a două matrice
- Caracteristici ale arhitecturilor sistolice