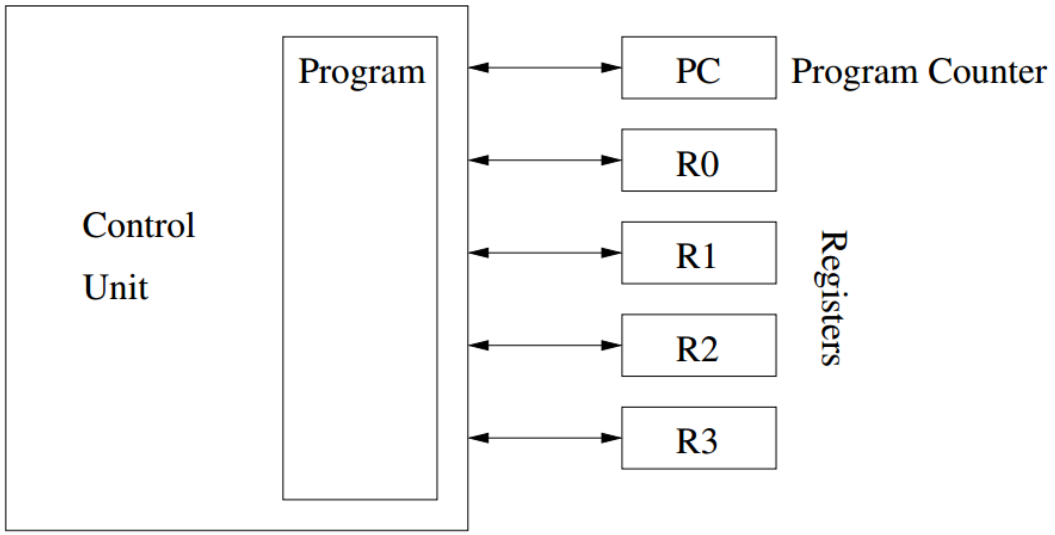
Modely paralelných výpočtov

Modely sekvenčných výpočtov

- **Turingov stroj**
- **Random Access Machine (RAM)**
  - pamäť = neohraničená postupnosť registrov
  - program = postupnosť príkazov (aktuálny príkaz určuje PC)
  - nepriama adresácia
  - podmienený skok
- **RASP – stored-program**

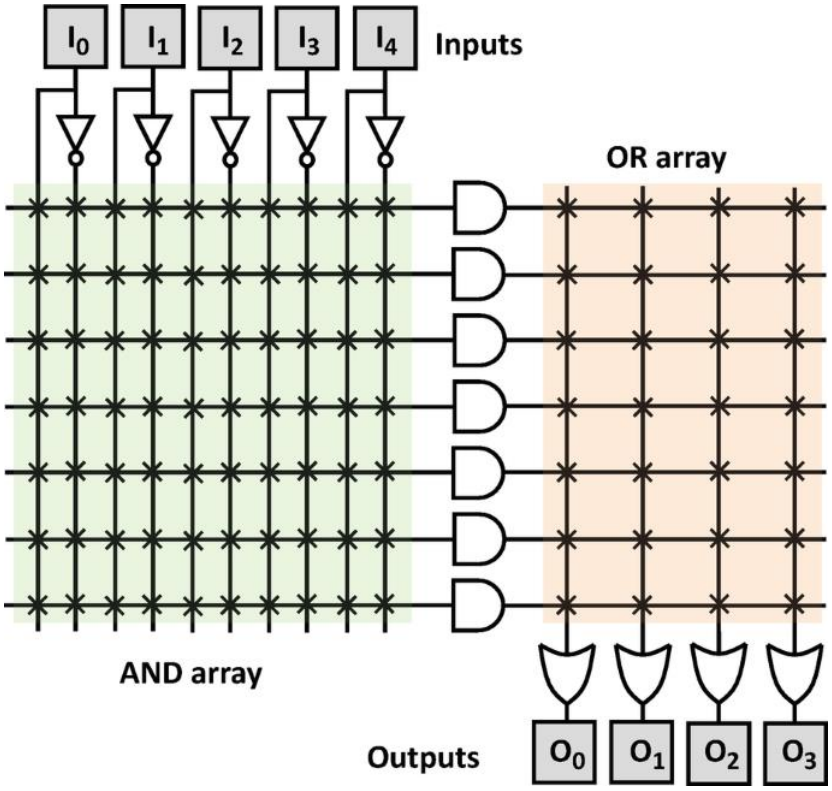


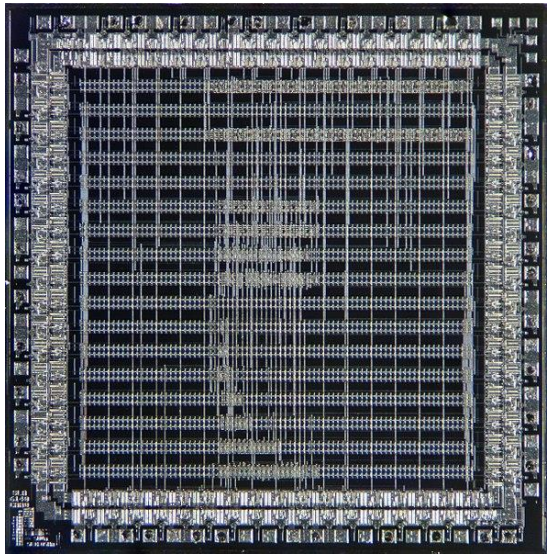
Modely paralelných výpočtov

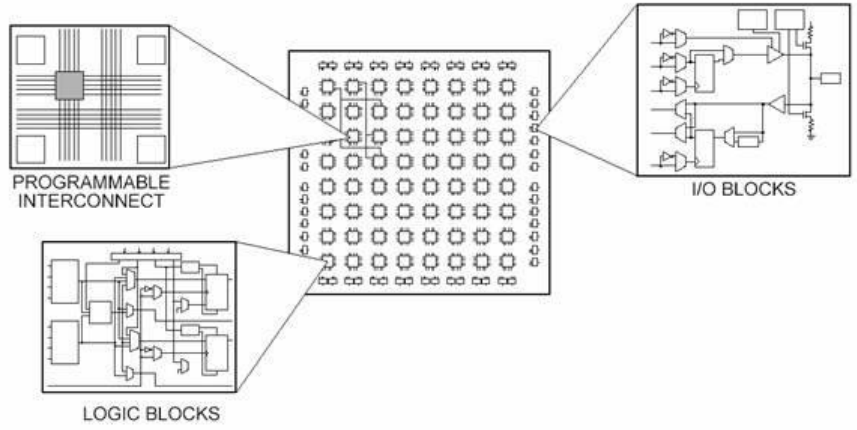
- **booleovské obvody**
- DAG – orientovaný acyklický graf
- sieťový model (systolické systémy)
  výpočty riadené údajmi (data-driven programming)
- modely so zdieľanou pamäťou
- ...

booleovské obvody

- návrh pomocou FPGA (Field-programmable gate array)
- konfigurovateľné logické bloky (polia blokov) prepojené komunikačnými kanálmi a I/O rozhraniami
- programovateľné v HDL (hardware description language) VHDL, Verilog ...
- OpenCL (Open Computing Language) univerzálne testovacie programovacie prostredie v štýle jazyka C
- FPGA Xilinx (AMD), Altera (Intel)
- masová výroba ASIC čipov (application-specific integrated circuits)


I0
I1
I2
I3
I4
Inputs
OR array
AND array
Outputs
O0
O1
O2
O3



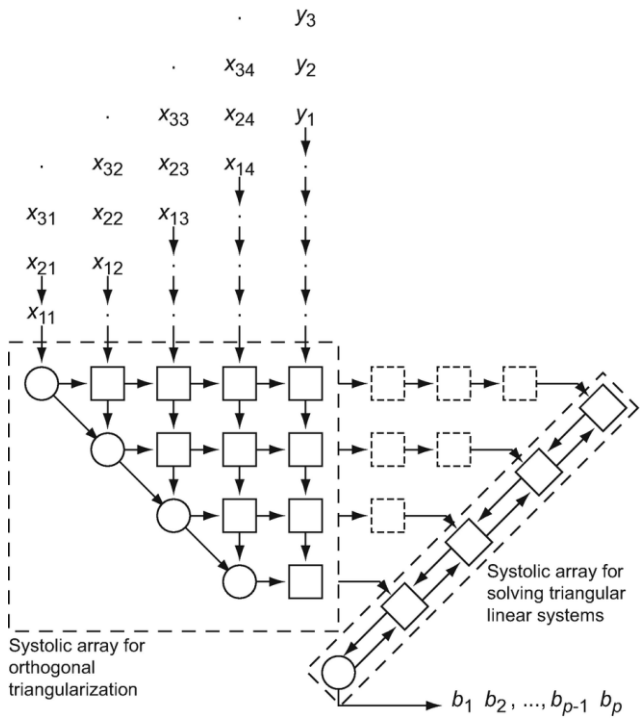

PROGRAMMABLE
INTERCONNECT
I/O BLOCKS
LOGIC BLOCKS

Modely paralelných výpočtov

- booleovské obvody
- **DAG – orientovaný acyklický graf – hľadanie kritickej cesty (Span)**
- **sieťový model (systolické systémy)**
  výpočty riadené údajmi (data-driven programming)
- modely so zdieľanou pamäťou
- ...

systolické pole

- sieť jednoduchých procesorov (transputerov), schopných v synchrónnom režime spracovať vstupy a poskytnúť ich na výstupe susedným procesorom
- jednoduché inštrukcie, minimálna lokálna pamäť možno hw implementovať
- výpočet riadený vstupmi



násobenie matíc v systolickom poli

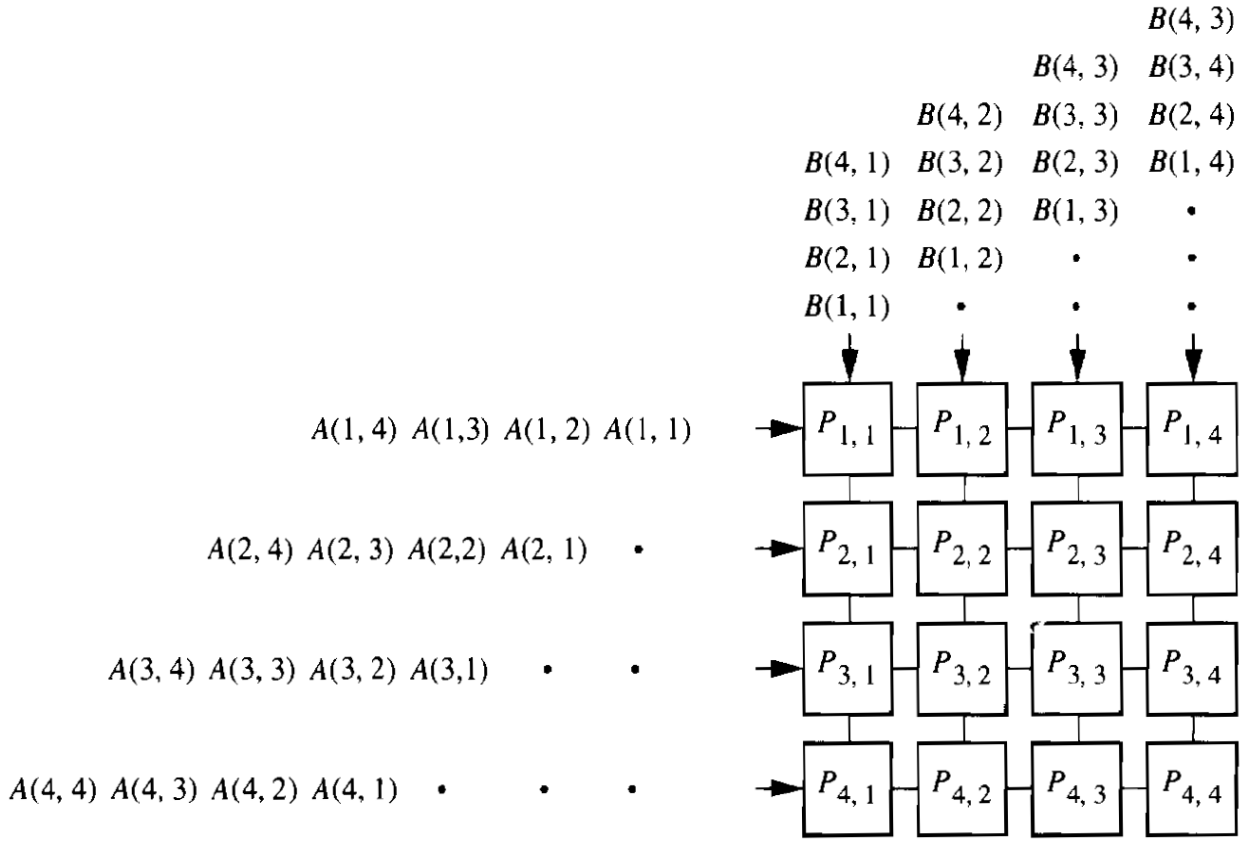
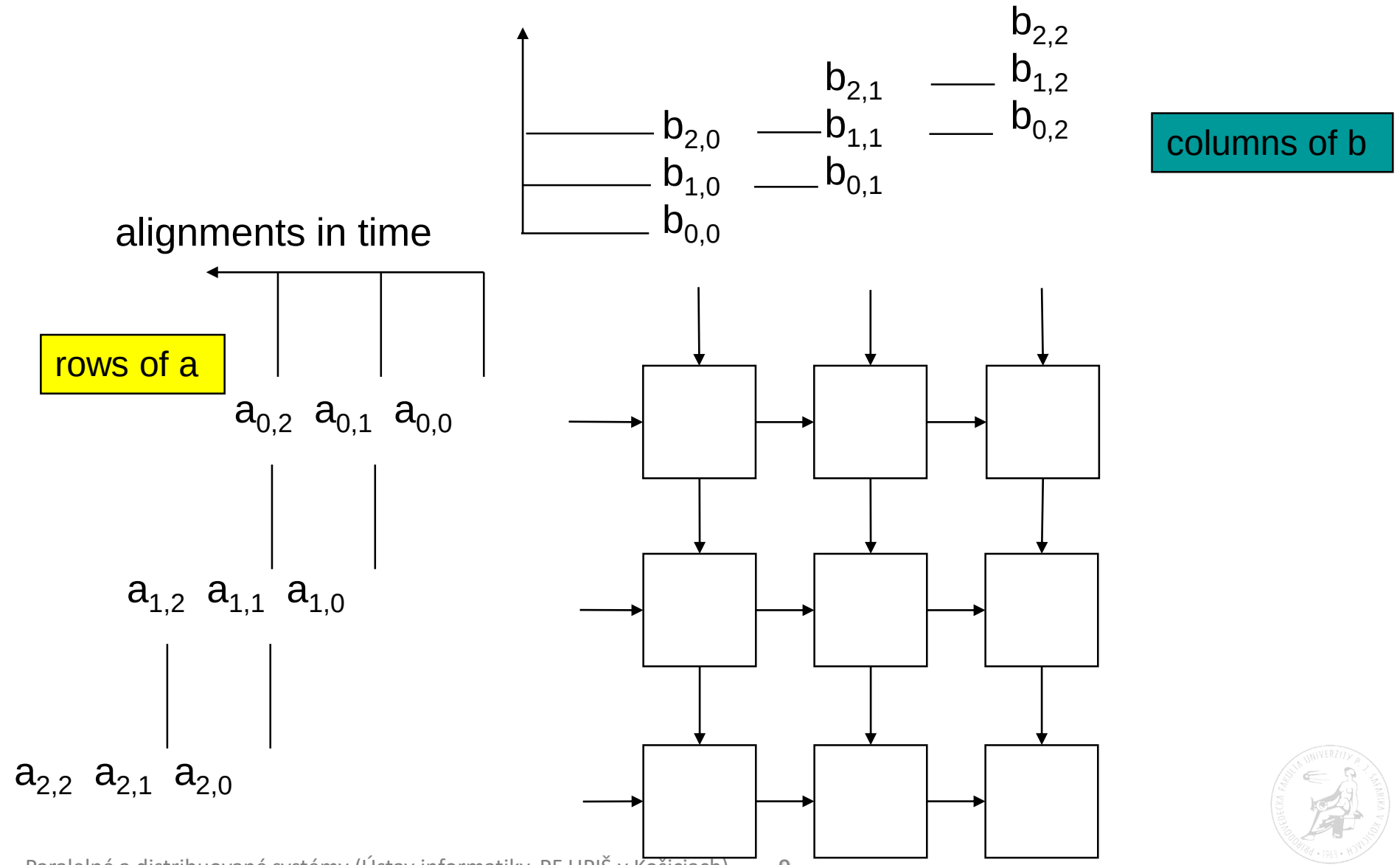
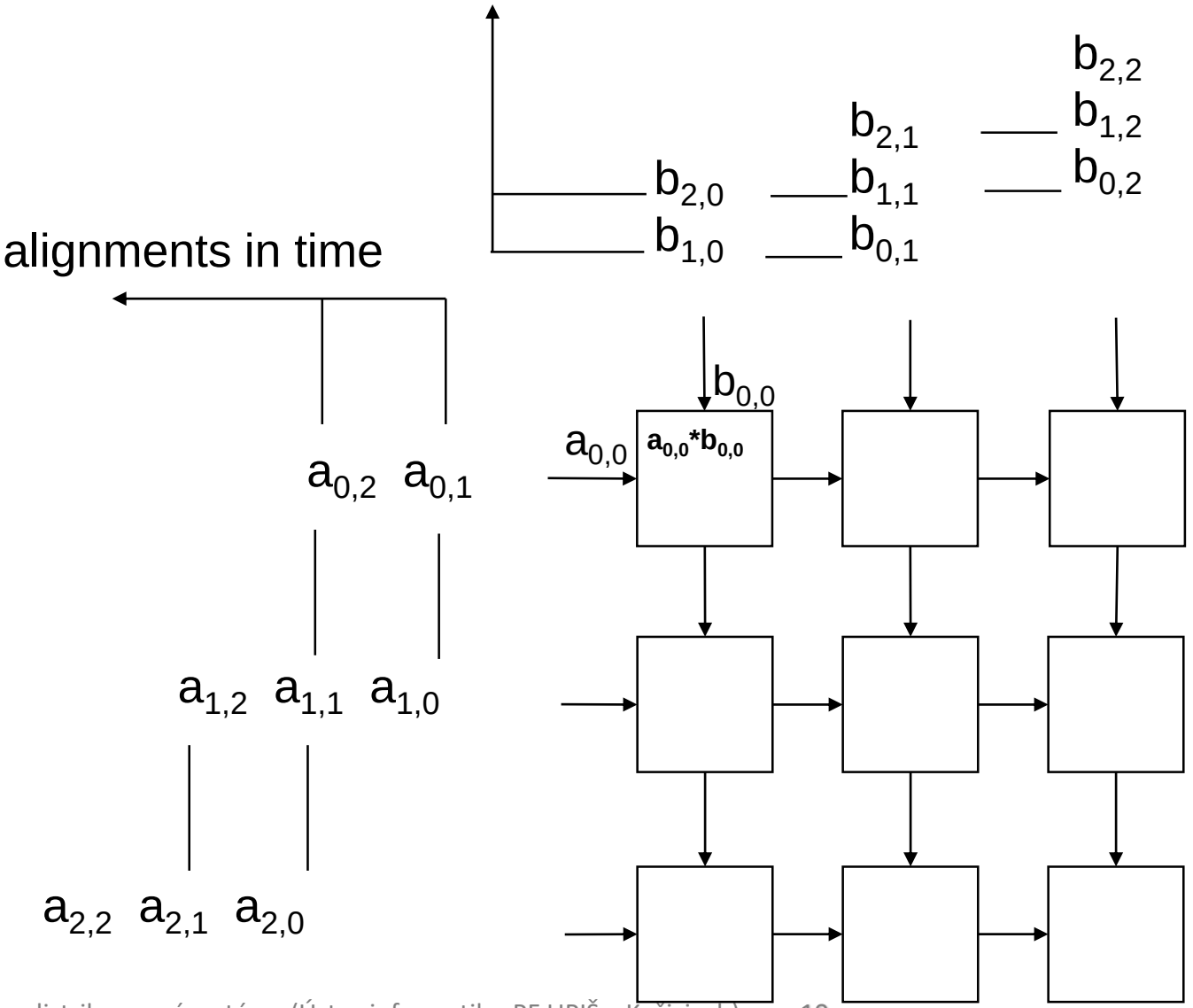


FIGURE 1.7
Matrix multiplication on the mesh using a systolic algorithm. Rows of A move synchronously into the left side, while columns of B move synchronously at the same rate into the top side. When $A(i, l)$ and $B(l, j)$ are available at processor $P_{i,j}$, the operation $C(i, j) = C(i, j) + A(i, l)B(l, j)$ takes place, $A(i, l)$ is sent to $P_{i,j+1}$ (if it exists), and $B(l, j)$ is sent to $P_{i+1,j}$ (if it exists).

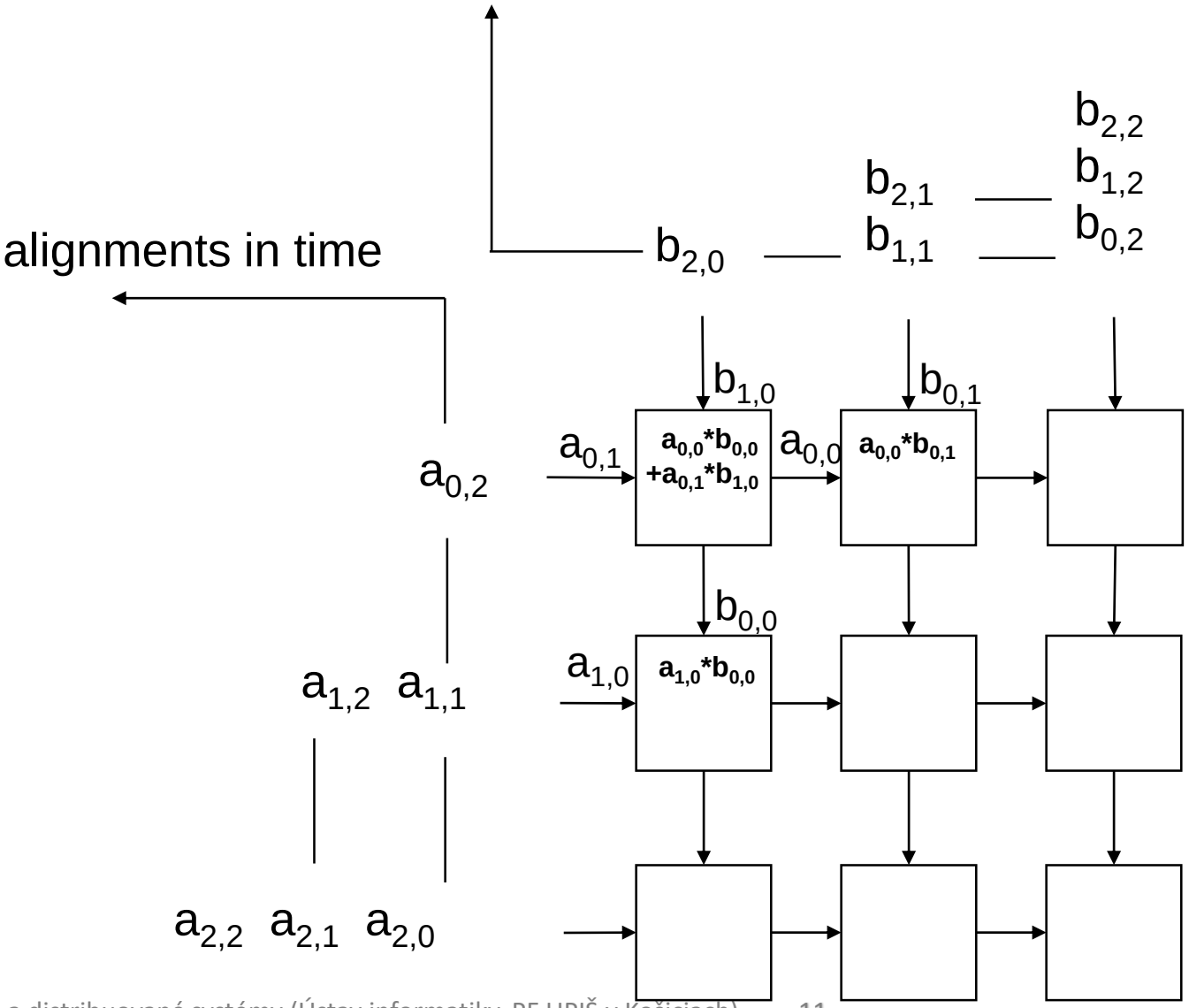
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



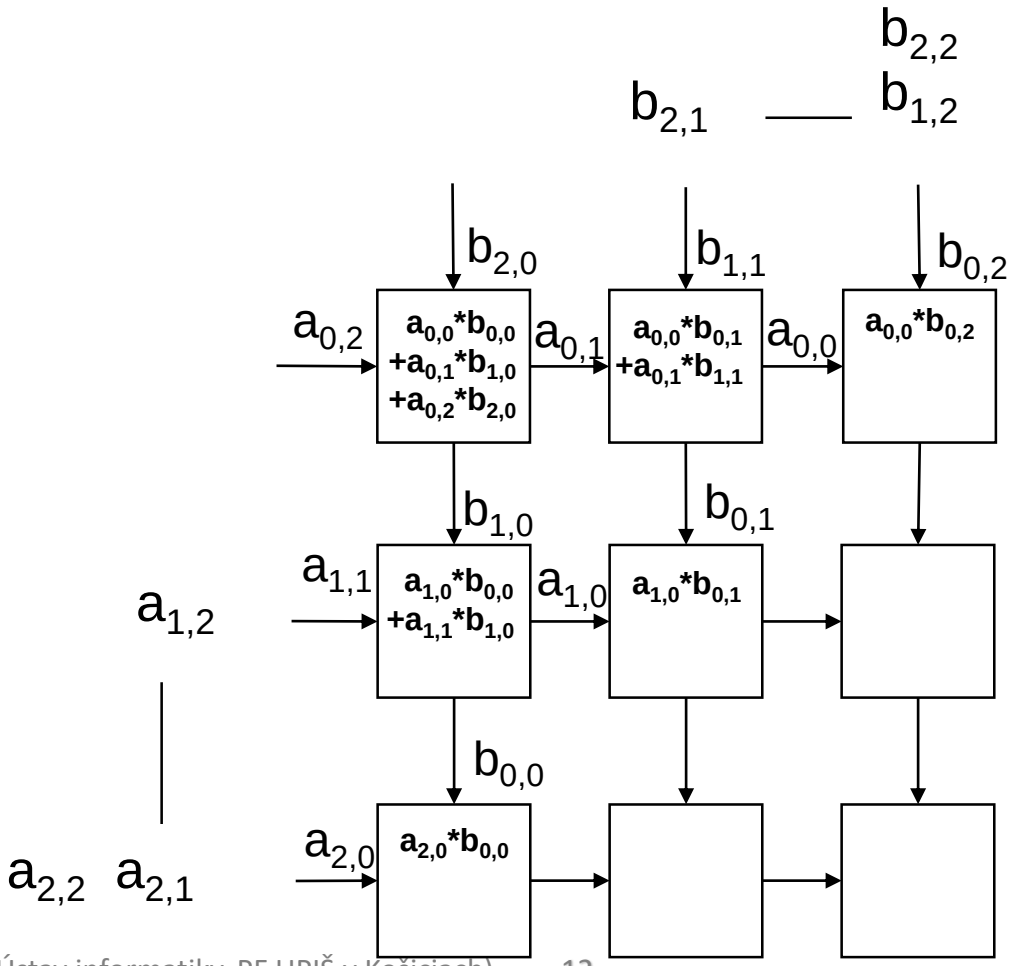
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



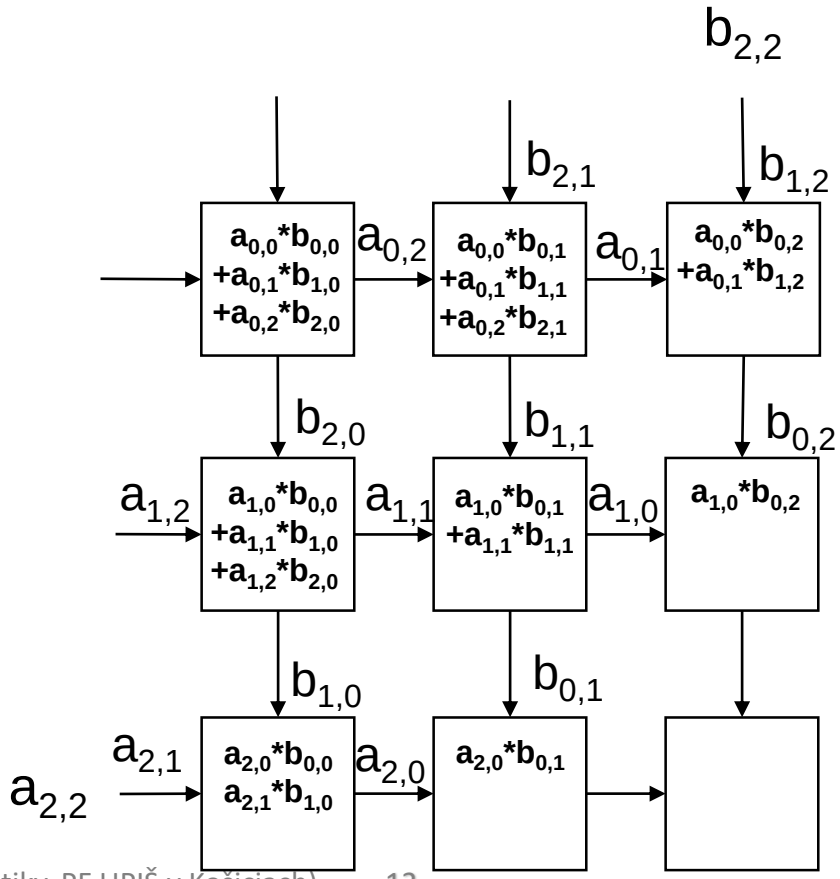
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



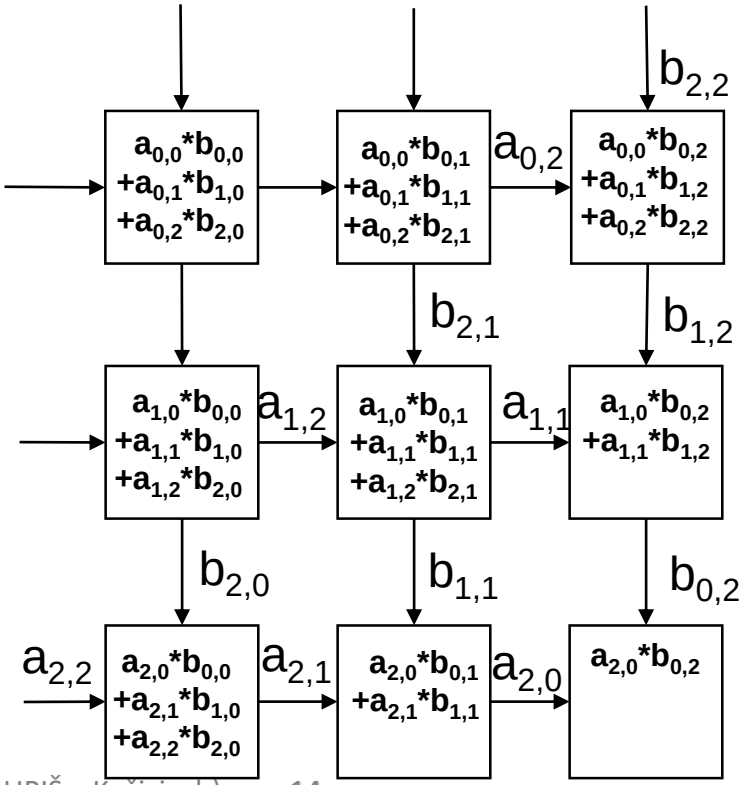
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



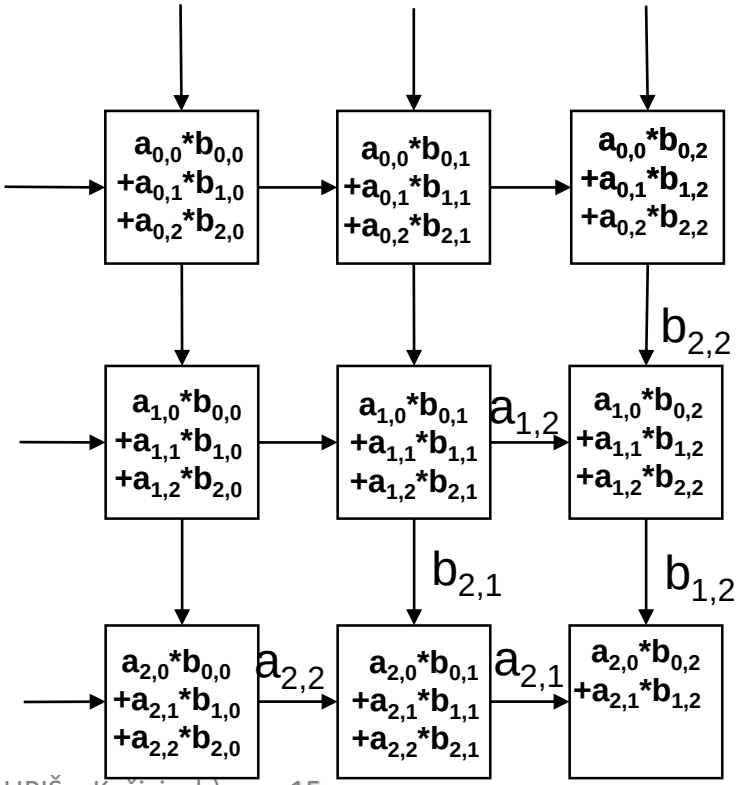
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



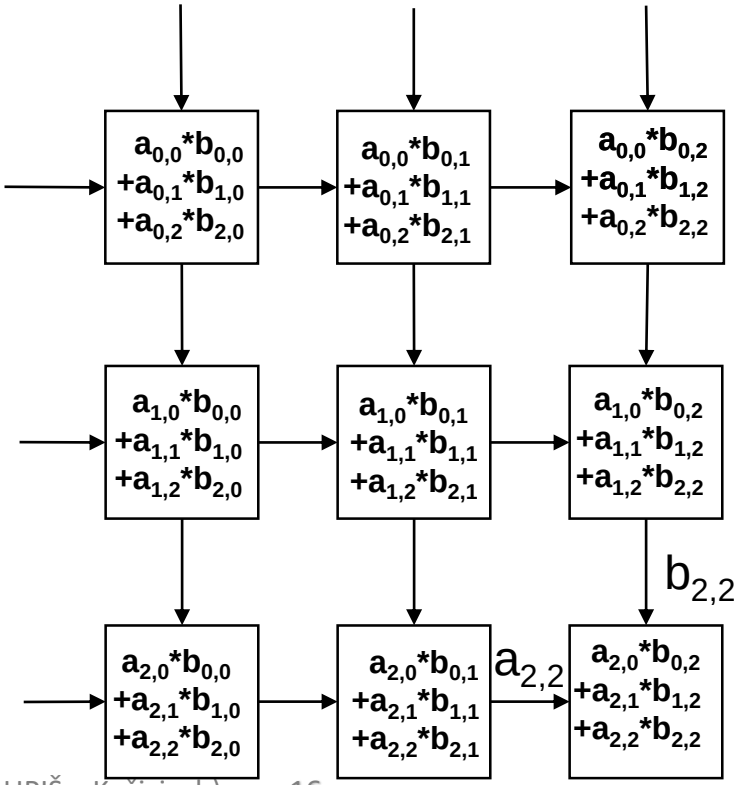
Systolic Matrix Multiplication - illustrated with two 3x3 matrices



Systolic Matrix Multiplication - illustrated with two 3x3 matrices

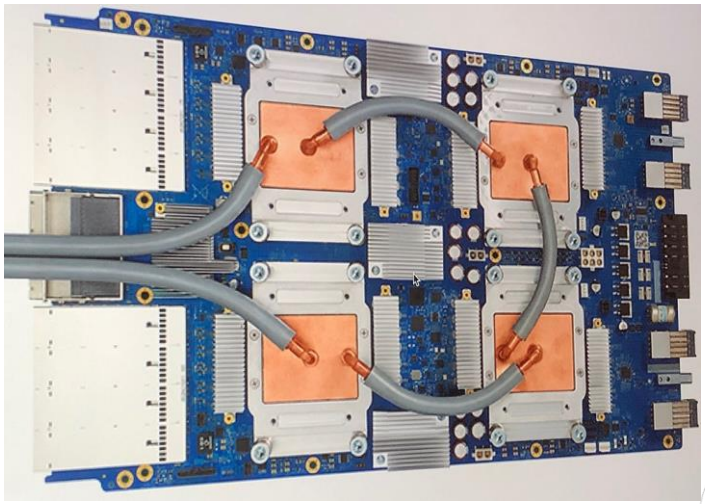


Systolic Matrix Multiplication - illustrated with two 3x3 matrices



paralelné spracovanie na úrovni hw

- zreťazenie spracovania inštrukcií (pipelining)
- out-of-order execution
- superscalar execution, VLIW
- TPU - hw podpora práce s vektormi a tenzormi pomocou systolických polí
- GPU – warps (SIMT)



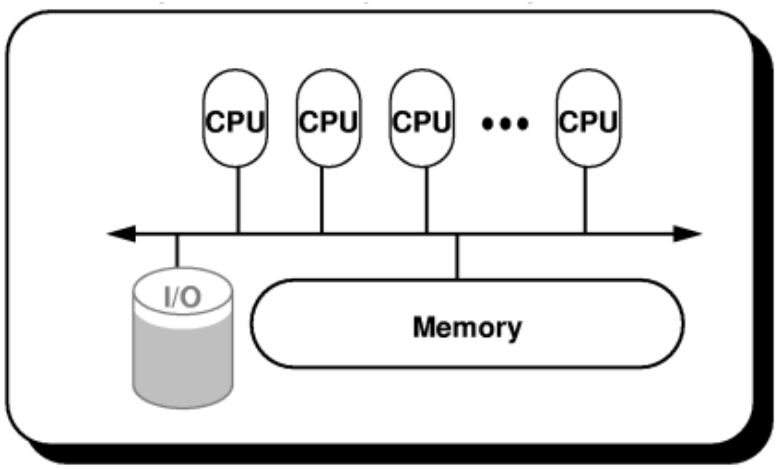
Paralelný počítač

- **Súbor** navzájom **prepojených procesorov** (obyčajne rovnakého typu) s cieľom **koordinovať** svoje aktivity a **vymieňať si údaje**.
  - predpokladá sa, že procesory sú umiestnene blízko seba
  - rôzne architektúry a režimy fungovania
    - architektúra (zbernica, prepínaná sieť ...)
    - počet procesorov
    - prístup k spoločnej pamäti
    - prepojenie a komunikačné protokoly
    - riadenie a synchronizácia

Modely paralelných výpočtov

- booleovské obvody
- DAG – orientovaný acyklický graf
- sieťový model (systolické systémy)
  výpočty riadené údajmi (data-driven programming)
- **modely so zdieľanou pamäťou**
- ...

Model so zdieľanou pamäťou



- Každý procesor má svoje **ID**
- Prístup do spoločnej pamäte:
  - **global_read**(Shared, Local)
  - **global_write**(Local, Shared)
- Sú procesory synchronizované (spoločné hodiny)?
  - synchrónny vs. asynchrónny model
- Beží na procesoroch rovnaký program?
  - **SIMD** – single instruction multiple data
  - **MIMD** – multiple instruction multiple data

PRAM model

- **PRAM** = **Parallel Random Access Machine**
  - model s (neobmedzenou) zdieľanou pamäťou
  - **synchrónny model** – spoločné hodiny
  - **SIMD** – v každom kroku sa vykonáva rovnaká inštrukcia
  - vstup aj výstup v zdieľanej pamäti
  - každý procesor má ID a pozná celkový počet procesorov
  - `if podmienka then`

    `príkaz;`

ak nie je podmienka splnená, procesor je príslušný počet krokov nečinný (idle)

Paralelné vyhľadávanie

- n procesorov P_1, P_2, …, P_n
- **Vstup**:
  - pole M[1..n] - obsahujúce zoznam n čísel
  - M[n+1] – hľadaná hodnota
- **Výstup**:
  - M[n+2] - index prvku poľa, ktoré obsahuje M[n+1]
- Kód pre procesor P_i:

```
global_read(M[n+1], x);
global_read(M[i], y);
if i=1 then global_write(0, M[n+2]);
if x=y then global_write(i, M[n+2]);
```

PRAM – prístup do pamäte

- rôzne varianty pri súčasnom prístupe na rovnaké miesto zdieľanej pamäte:

  EREW PRAM – Exclusive Read, Exclusive Write

  CREW PRAM – Concurrent Read, Exclusive Write

  CRCW PRAM – Concurrent Read, Concurrent Write

  - Common – pri zápise sa zhodujú hodnoty
  - Arbitrary – uspeje ľubovoľný
  - Priority – uspeje procesor s najnižším indexom

- reálne je prístup do spoločnej pamäte vždy pomalší, ako prístup do lokálnej pamäte

Paralelné vyhľadávanie

- n procesorov P_1, P_2, …, P_n
- **Vstup**:
  - pole M[1..n] - obsahujúce zoznam n čísel
  - M[n+1] – hľadaná hodnota
- **Výstup**:
  - M[n+2] - index prvku poľa, ktoré obsahuje M[n+1]
- Kód pre procesor P_i:

```
global_read(M[n+1], x);                    CR
global_read(M[i], y);                      ER
if i=1 then global_write(0, M[n+2]);       EW
if x=y then global_write(i, M[n+2]);       CW-arbitrary
```

CRCW-arbitrary

Výkon paralelných algoritmov (1)

- p – počet procesorov
- $T^*(n)$ – sekvenčný čas – počet krokov optimálneho sekvenčného algoritmu
- $T_p(n)$ – paralelný čas pri p procesoroch
- Span: $T_\infty(n)$
- Zrýchlenie (speed-up):

$$S_p(n) = T^*(n) / T_p(n)$$

- Cena (cost):

$$C_p(n) = p.T_p(n)$$

Výkon paralelných algoritmov (2)

- Cenovo optimálny algoritmus (cost-optimal):

$$C_p(n) \in O(T^*(n))$$

- Efektívnosť:

$$E_p(n) = T^*(n)/C_p(n) = S_p(n)/p$$

- Dôsledky: $S_p(n) \leq p$ (prečo?), $E_p(n) \leq 1$
- Zrýchlenie:
  - $S_p(n) \sim p$ – lineárne zrýchlenie
  - $S_p(n) > p$ – super-lineárne zrýchlenie (dá sa niekedy dosiahnuť vďaka rôznym pamäťovým modelom)

Paralelné sčítanie (1)

- **Vstup:** pole A[1..n], kde n = 2^k
- **Výstup:** súčet prvkov poľa A v premennej S
- Sekvenčný algoritmus v čase O(n)

```
S := A[1]; for i := 2 to N do S := S + A[i];
```

- Paralelný algoritmus pre **n** procesorov

Kód pre procesor i (1 ≤ i ≤ n):

```
global_read(A[i], a);   ER
global_read(S, x);      CR
x := x + a;
global_write(x, S);     CW ??
```

race condition !

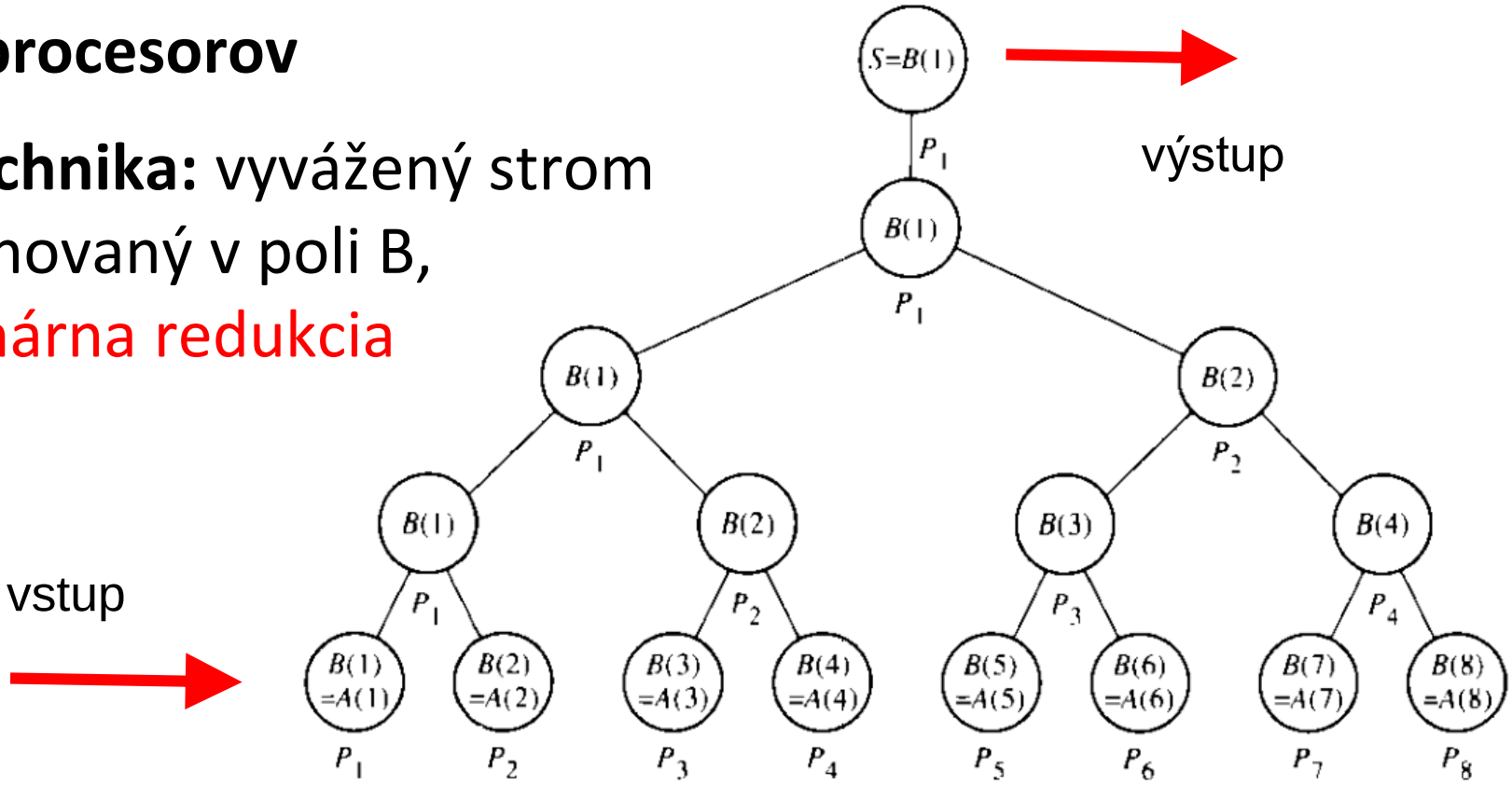
Paralelné sčítanie (1)

- **Vstup:** pole A[1..n], kde n = 2^k
- **Výstup:** súčet prvkov poľa A v globálnej premennej S
- **n procesorov**
- **Rekurzívne**
  súčet prvej polovice + súčet druhej polovice
  S (A[1..n]) = S(A[1..n/2]) + S(A[n/2+1..n]
  polovica procesorov vypočíta prvý súčet, polovica druhý
  rekurzia ...

Kód pre procesor i (1 ≤ i ≤ n): ?

Paralelné sčítanie (1)

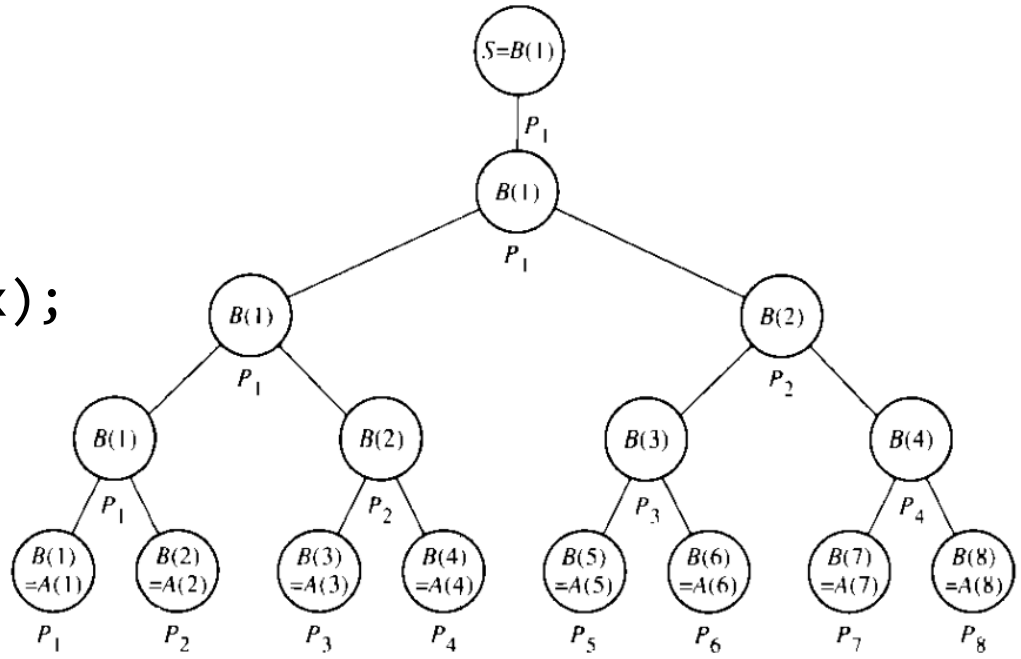
- **Vstup:** pole A[1..n], kde n = 2^k
- **Výstup:** súčet prvok poľa A v globálnej premennej S
- **n procesorov**
- **Technika:** vyvážený strom uchovaný v poli B, binárna redukcia



Paralelné sčítanie (2)

Kód pre procesor i ($1 \leq i \leq n$):

```
global_read(A[i], a);
global_write(a, B[i]);
for h:=1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(B[2*i-1], x);
     global_read(B[2*i], y);
     z := x + y;
     global_write(z, B[i]);
   end;
if i = 1 then global_write(z, S);
```



Paralelné sčítanie (3)

Kód pre procesor i ($1 \leq i \leq n$):

```
global_read(A[i], a);
global_write(a, B[i]);
for h:=1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(B[2*i-1], x);
     global_read(B[2*i], y);
     z := x + y;
     global_write(z, B[i]);
   end;
if i = 1 then global_write(z, S);
```

EREW model

$T_n(n) \in \Theta(\log(n))$

$S_n(n) \in \Theta(n/\log(n))$

$C_n(n) \in \Theta(n.\log(n))$

$E_n(n) \in \Theta(1/\log(n))$

nie je cenovo optimálny

Paralelné sčítanie (4)

- Škálovateľnosť riešenia pre p procesorov, kde p < n:
  1. každý procesor dostane segment s n/p číslami a vypočíta jeho súčet sekvenčne na n/p krokov
  2. binárnou redukciou p procesorov spočíta paralelne súčet na log p krokov
- Analýza:
  - $T_p(n) \in \Theta(n/p + \log p)$
  - $C_p(n) \in \Theta(p.(n/p + \log p)) = \Theta(n + p.\log p)$
    (optimum pri p = n/(log n))
  - $S_p(n) \in \Theta(n.p / (n + p.\log p))$
  - $E_p(n) \in \Theta(n / (n + p.\log p))$

Paralelné sčítanie – aplikácie

- Namiesto + môžeme uvažovať ľubovoľnú binárnu asociatívnu operáciu (*, min, max, …)

- **Úloha:** Koľkokrát sa zadaná hodnota nachádza v poli?
  - Vstup: pole A[1..n], hľadaná hodnota h
  - Pomocné pole: B[1..n]
  - B[i] = 0 ak A[i] <> h
  - B[i] = 1 ak A[i] = h
  - B[1] + B[2] + ... + B[n] = počet výskytov h

Paralelné sčítanie s modifikáciou vstupu

Kód pre procesor i ($1 \leq i \leq n$ ... stačí $n/2$):

```
global_read(A[i], x);
for h:=1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(A[n/2^h + i], y);
     x := x + y;
     global_write(x, A[i]);
   end;
if i = 1 then global_write(x, S);
```

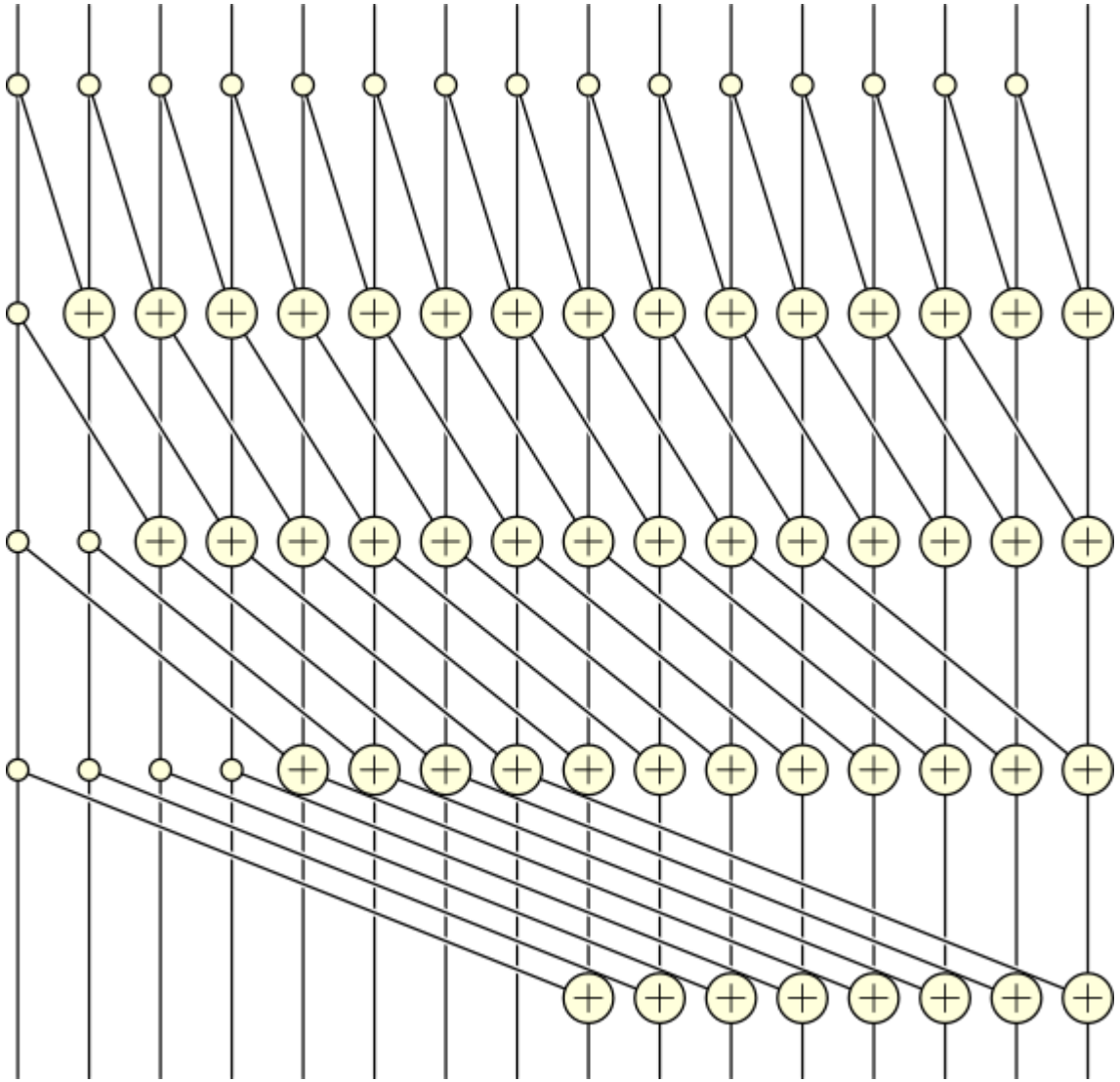
pre $n \neq 2^k$?

Prefixové súčty

- **Vstup:**
  - binárna asociatívna operácia + (*, min, max, or, …)
  - n-prvkové pole A, kde $n = 2^k$
- **Výstup**:
  - n-prvkové pole S také, že S[i] = A[1] + A[2] + … + A[i]
- **Sekvenčný algoritmus v čase O(n):**

```
S[1] := A[1];
for i := 2 to N do
  S[i] := S[i-1] + A[i];
```

Hillis-Steele prefix sum



Hillis-Steele prefix sum

Hillis-Steelov algoritmus na výpočet prefixových súčtov
kód pre procesor i ($1 \leq i \leq n$):

```
global_read(A[i], x);
global_write(x, S[i]);
for h := 0 to log(n)-1 do
    if i > 2^h then
    begin
        global_read(S[i-2^h], y);
        x := x + y;
        global_write(x, S[i]);
    end;
```

EREW model

$T^*(n) = n$

$T_n(n) \in \Theta(\log(n))$

$S_n(n) \in \Theta(n/\log(n))$

$C_n(n) \in \Theta(n.\log(n))$

$E_n(n) \in \Theta(1/\log(n))$

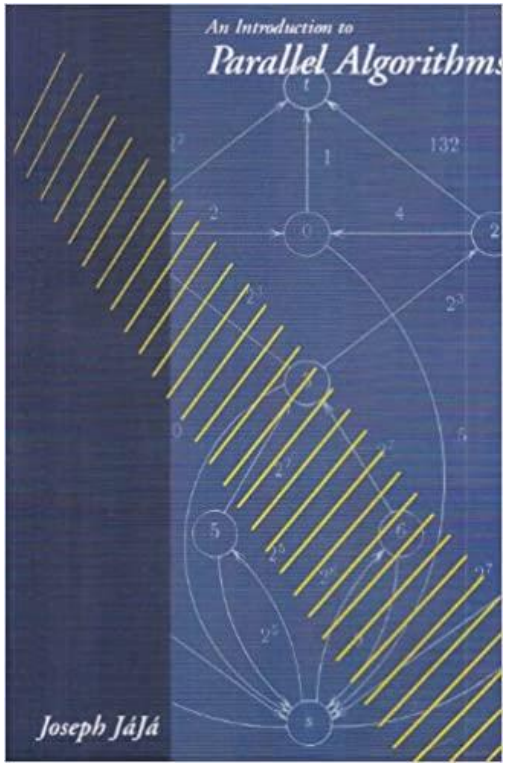
prefixové súčty – aplikácie

- namiesto + môžeme uvažovať ľubovoľnú binárnu asociatívnu operáciu (*, min, max, …)
- histogramy, radix sort, paralelná aritmetika, interpolácie
- **Úloha:** Pre každý index i poľa určiť, koľkokrát sa hodnota h nachádza v podpoli 1,...,i.
  - Vstup: pole A[1..n], hľadaná hodnota h
  - Pomocné pole: B[1..n]
  - B[i] = 0 ak A[i] <> h
  - B[i] = 1 ak A[i] = h
  - B[1] + B[2] + ... + B[i] = počet výskytov h v podpoli A[1..i]

študijná literatúra

- Joseph Jájá - An Introduction to Parallel Algorithms

ISBN-13: 978-0201548563
ISBN-10: 0201548569



Ďakujem za pozornosť !