

Paralelní systémy

Protočnost (pipelining)

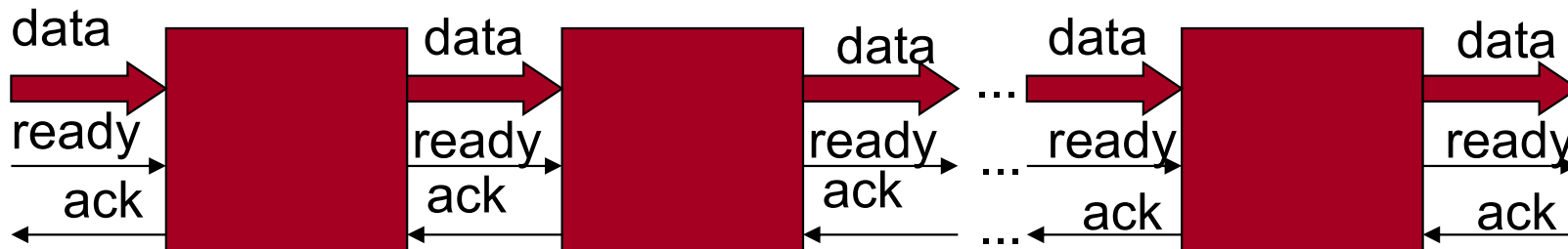
Protočnost (pipelining)

* Protočnost je tehnika projektovanja hardvera kojom se uvodi konkurentnost u računarski sistem tako što se neke osnovne funkcije (f) čije se izvršenje često zahteva dele na niz podfunkcija f_1, f_2, \dots, f_k , tako da budu zadovoljeni sledeći kriterijumi:

- Izračunavanje osnovne funkcije f je ekvivalentno sekvencijalnom izračunavanju podfunkcija f_1, f_2, \dots, f_k .
- izlazi prethodne podfunkcije predstavljaju ulaze za sledeću podfunkciju u nizu podfunkcija koje se izvršavaju
- Osim razmene podataka između podfunkcija ne postoji nikakva druga zavisnost
- Može se projektovati hardver za izračunavanje svake podfunkcije
- Vremena potrebna ovim hardverskim jedinicama da obave individualna izračunavanja su približno jednaka

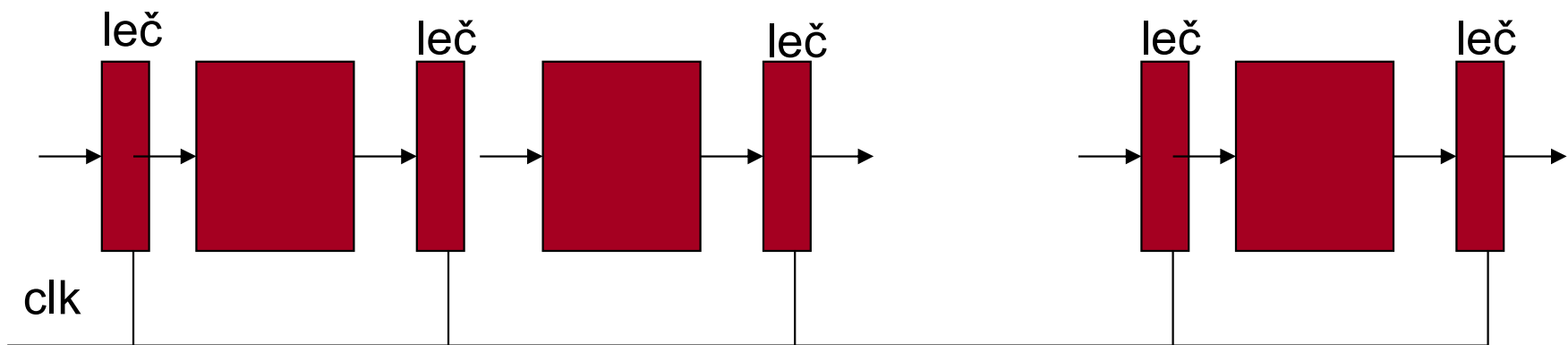
Protočnost (nast.)

- * Hardver za izračunavanje bilo koje podfunkcije zove se stepen protočnog sistema (pipeline stage)
- * U zavisnosti od načina upravljanja tokom podataka kroz protočni sistem mogu se razlikovati
 - asinhroni protočni sistemi
 - sinhroni protočni sistemi
- * Asinhroni model – razmenom podataka izmedju dva susedna stepena upravlja se nekom handshake procedurom. Hardverski stepeni sadrže memorijske elemente



Protočnost (nast.)

- * Sinhroni model – razmenom podataka upravlja se pomoću globalnog clk. Hardverski stepeni ne sadrže memorijske elemente. Zato se izmedju stepena ubacuju lečevi.



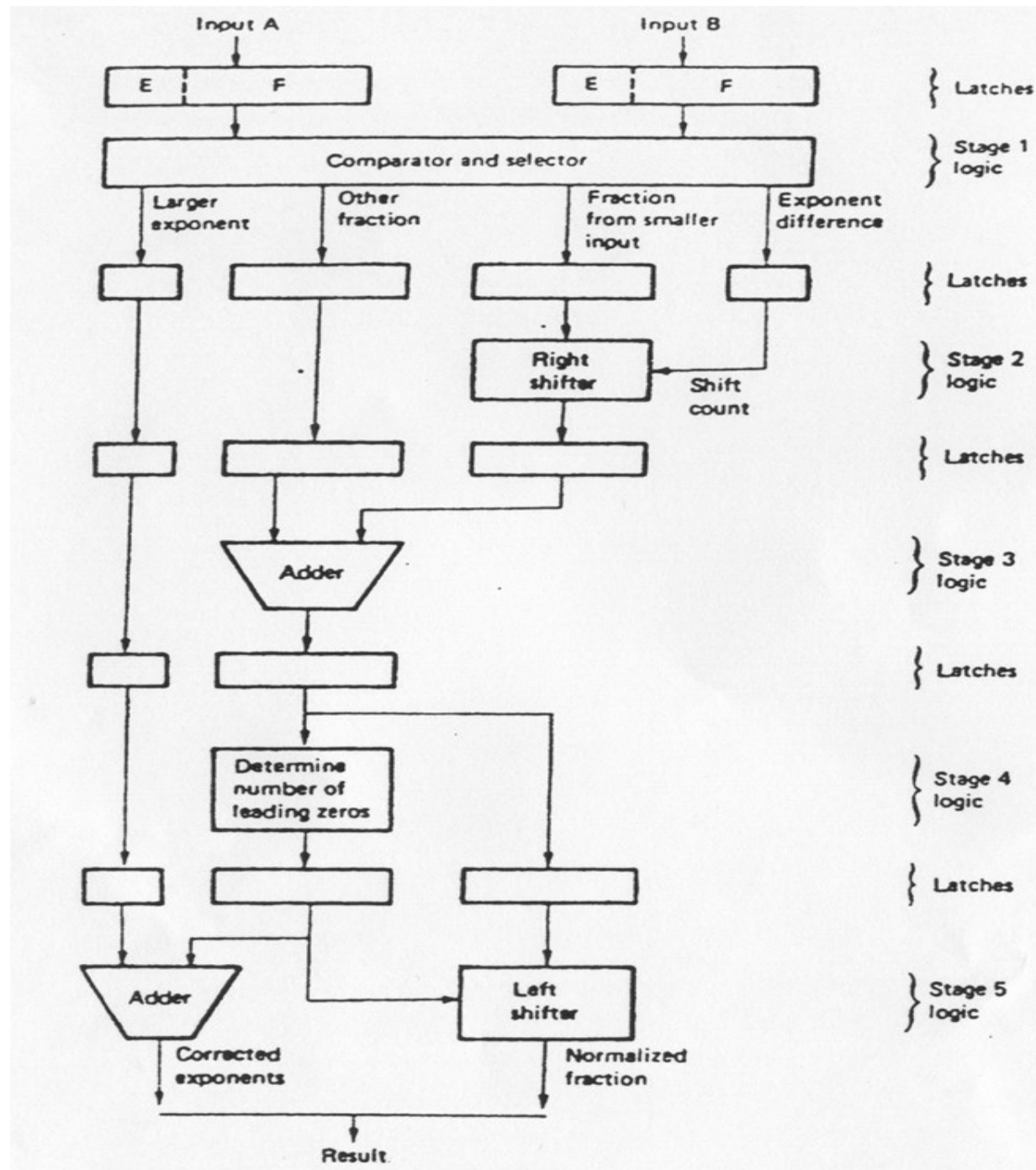
- svi stepeni su aktivni u svakom klok ciklusu. Stepen i unosi kašnjenje T_i . Klok perioda protočnog sistema iznosi
- $T = \max\{T_1, T_2, \dots, T_k\} + T_L$, gde je T_L kašnjenje koje unosi leč

Primer: projektovanje protočnog FP sabirača

* $A = a * 2^p$, $B = b * 2^q$

- * korak1: Poredjenje eksponenata p i q da bi se pronašao veći, $r = \max(p, q)$ i razlika $t = |p - q|$.
- * korak2: Pomeriti za t mesta u desno mantisu manjeg broja da bi se izjednačili eksponenti pre sabiranja
- * korak3: Sabiranje mantisa i dobijanje medjurezultata
- * korak4: Odredjivanje broja vodećih nula u sumi, recimo u .
- * korak5: Pomeranje dobijene sume za u mesta u levo da bise dobila normalizovana mantisa i ažuriranje većeg eksponenta: $r + u$

Protoční sabirač



Protočni sabirač

* Neaka su kašnjenja koja unose pojedini stepeni

- $T_1 = 60 \text{ ns}$
- $T_2 = 50 \text{ ns}$
- $T_3 = 80 \text{ ns}$
- $T_4 = 50 \text{ ns}$
- $T_5 = 80 \text{ ns}$
- $T_L = 10 \text{ ns}$

* Klok perioda protočnog sistema je $T = \max\{60, 50, 80, 50, 50\} + 10 = 90 \text{ ns}$

* Vreme potrebno neprotočnom sabiraču da sabere dva FP broja iznosi $T_{np} = 60 + 50 + 80 + 50 + 80 = 320 \text{ ns}$

* Vreme potrebno protočnom sabiraču da sabere dva FP broja iznosi $T_{pr} = 5 * 90 = 450 \text{ ns}$

Gde je dobit od uvođenja protočnosti?

* Ako je potrebno sabrati n parova brojeva neprotočnom sabiraču će biti potrebno

- $T_{np} = n * 320 \text{ ns}$

* a protočnom

- $T_{pr} = 450 + (n-1) * 90 \text{ ns}$

* Za $n=10$

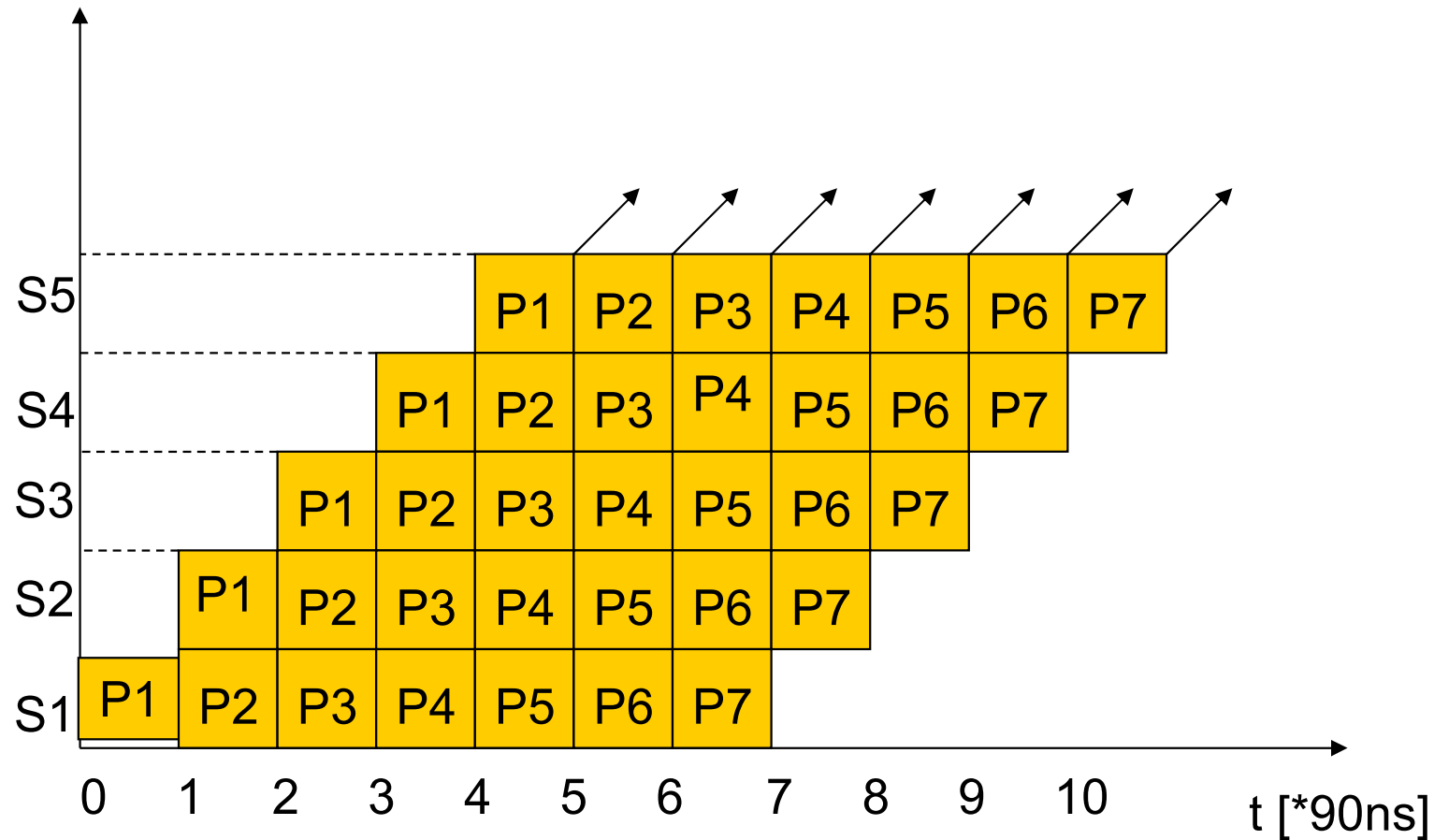
- $T_{np} = 10 * 320 = 3200 \text{ ns}$

- $T_{pr} = 450 + 9 * 90 = 1360 \text{ ns}$

* Što je veće n performanse protočnog sistema su bolje. Za dovoljno veliko n ubrzanje protočnog sistema jednako je broju stepena, k .

Gantov dijagram

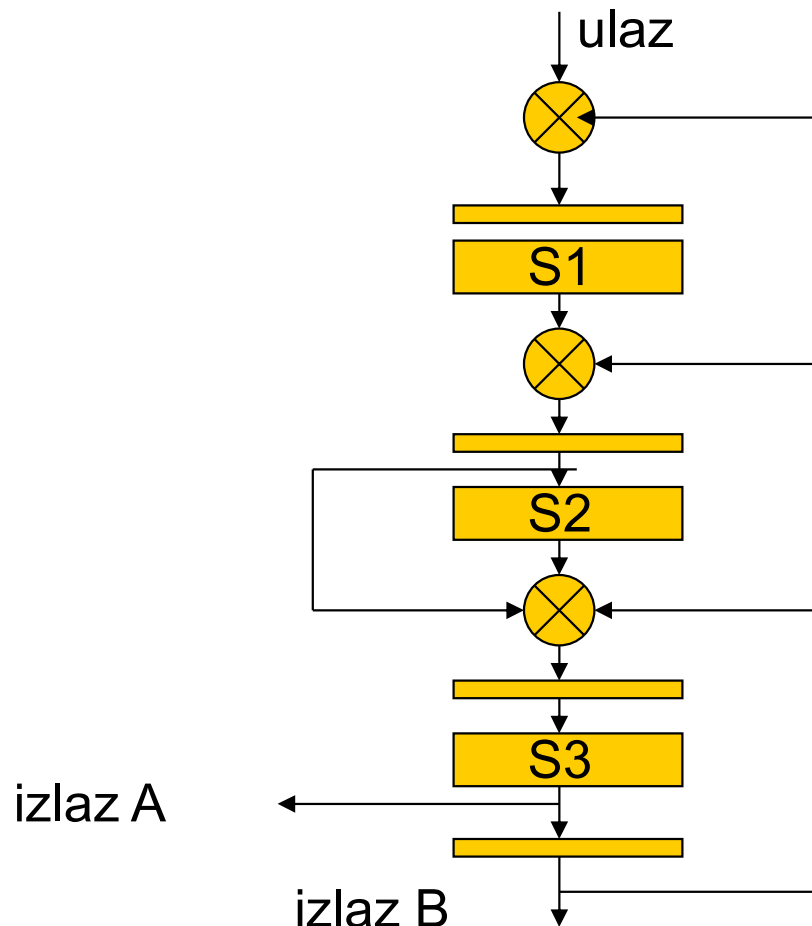
* Prikazuje zauzetost pojedinih stepena u vremenu



Klasifikacija protočnih sistema

* U odnosu na način povezivanja hardverskih stepena:

- Linearni (kaskadna veza izmedju stepena; sabirač iz prethodnog primera)
- Nelinearni – pored kaskadnih veza postoje veze izvedene u napred i povratne (u nazad)



Klasifikacija protočnih sistema

* U odnosu na mogućnosti obrade

- Jednofunkcijski – protočni sistemi sa fiksno dodeljenom funkcijom (sabirač iz prethodnog primera)
- Višefunkcijski – mogu obavljati više funkcija u isto ili različitim vremenskim trenucima. Mogu biti
 - Statički
 - Dinamički

* Protočnost na mikro nivou:

- kod savremenih računara koristi na nivou:
 - izvršenja instrukcija
 - izvršenja ALU operacija
 - kod pristupa memoriji

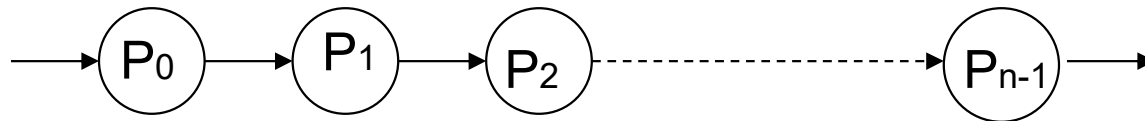
* Protočno na nivou zadatka (na makro nivou)

Primer – sortiranje niza

* Zadat je niz od n brojeva koji treba sortirati u datom redosledu (npr. rastućem)

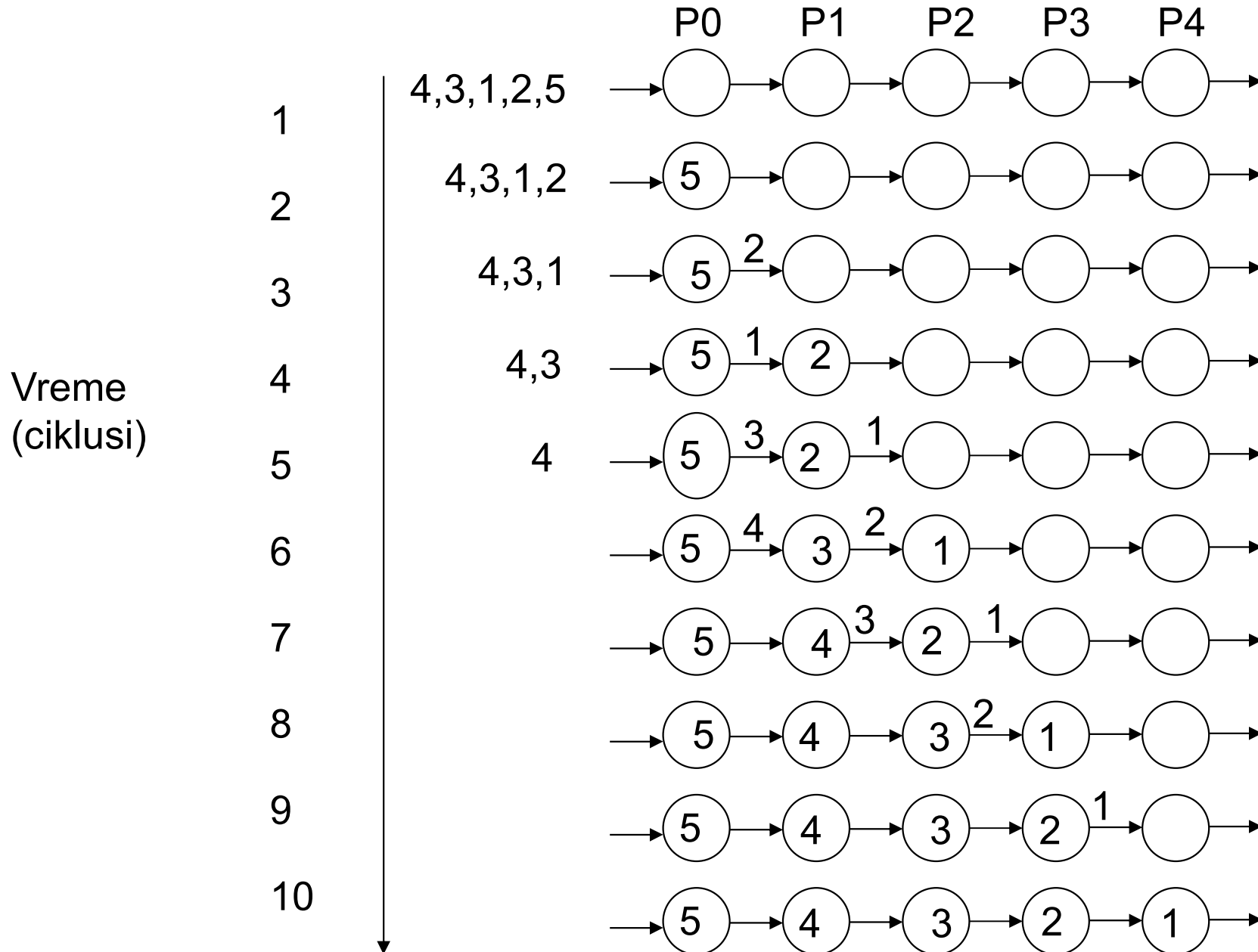
* Protočna realizacija:

- Protočni sistem od n procesnih jedinica povezanih linearno



- Procesor P_0 prihvata niz brojeva (po jedan u datom klok ciklusu)
 - Pamti najveći do tog trenutka primljeni broj
 - Prosledjuje desnom susedu manji broj
 - Svaki procesni element P_i u nizu obavlja istu operaciju (pamti najveći primljeni broj a prosledjuje manji ka P_{i+1})
 - Na kraju rada P_0 će imati najveći broj, P_1 sledeći najveći, ... P_{n-1} će imati najmanji broj
- Algoritam je paralelna verzija insertion sort algoritma

Primer n=5



Primer (nast.)

* osnovni program za procesor P_i

```
primi(broj,  $P_{i-1}$ )  
If (broj > x) {posalji(x,  $P_{i+1}$ );  
x=broj;}  
else posalji(broj,  $P_{i+1}$ );
```

* Ako ima n brojeva i n procesora, i -ti procesor ($i=0,1,\dots,n-1$) treba da primi $n-i$ brojeva, a da prosledi $n-i-1$, jer se jedan od brojeva zadržava.

* Kompletan program za P_i

```
Broj_desnih_procesora= $n-i-1$ ;  
Primi(x,  $P_{i-1}$ );  
For (j=0: j < Broj_desnih_procesora; j++) {  
  primi(broj,  $P_{i-1}$ )  
  If (broj > x) {posalji(x,  $P_{i+1}$ );  
  x=broj;}  
  else posalji(broj,  $P_{i+1}$ );  
}
```

Analiza performansi

* Sekvencijalni algoritam

- Usvojimo da poredjenje i dodeljivanje vrednosti traju 1 vremensku jedinicu;
- Sekvencijalno vreme izvršenja
 - $T_1 = (n-1) + (n-2) + \dots + 2 + 1 = n(n-1)/2 \approx O(n^2/2)$

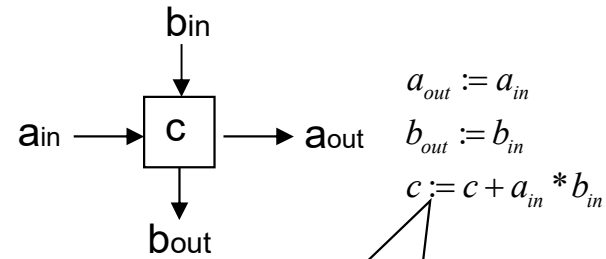
* Paralelni algoritam

- Paralelna implementacija ima $n + n - 1 = 2n - 1$ ciklusa, ako ima n procesora i n brojeva koje treba sortirati
 - Svaki procesor u u jednom ciklusu obavlja poredjenje i prijem i predaju podataka (izuzev poslednjeg)
 - $T_{\text{ciklusa}} = 1 + t_{\text{komunikacije}}$
 - Potrebno je još n ciklusa da se protočni sistem isprazni
 - Ukupno vreme rada protočnog sistema iznosi
 - $T_p = (3n - 1) T_{\text{ciklusa}}$

● Ubrzanje

$$S = \frac{T_1}{T_p} = \frac{\frac{n(n-1)}{2}}{3n-1} = \frac{n(n-1)}{2(3n-2)} = O\left(\frac{n}{6}\right)$$

Primer: Množenje
Matrica $C=A*B$, $A_{3 \times 3}$, $B_{3 \times 3}$

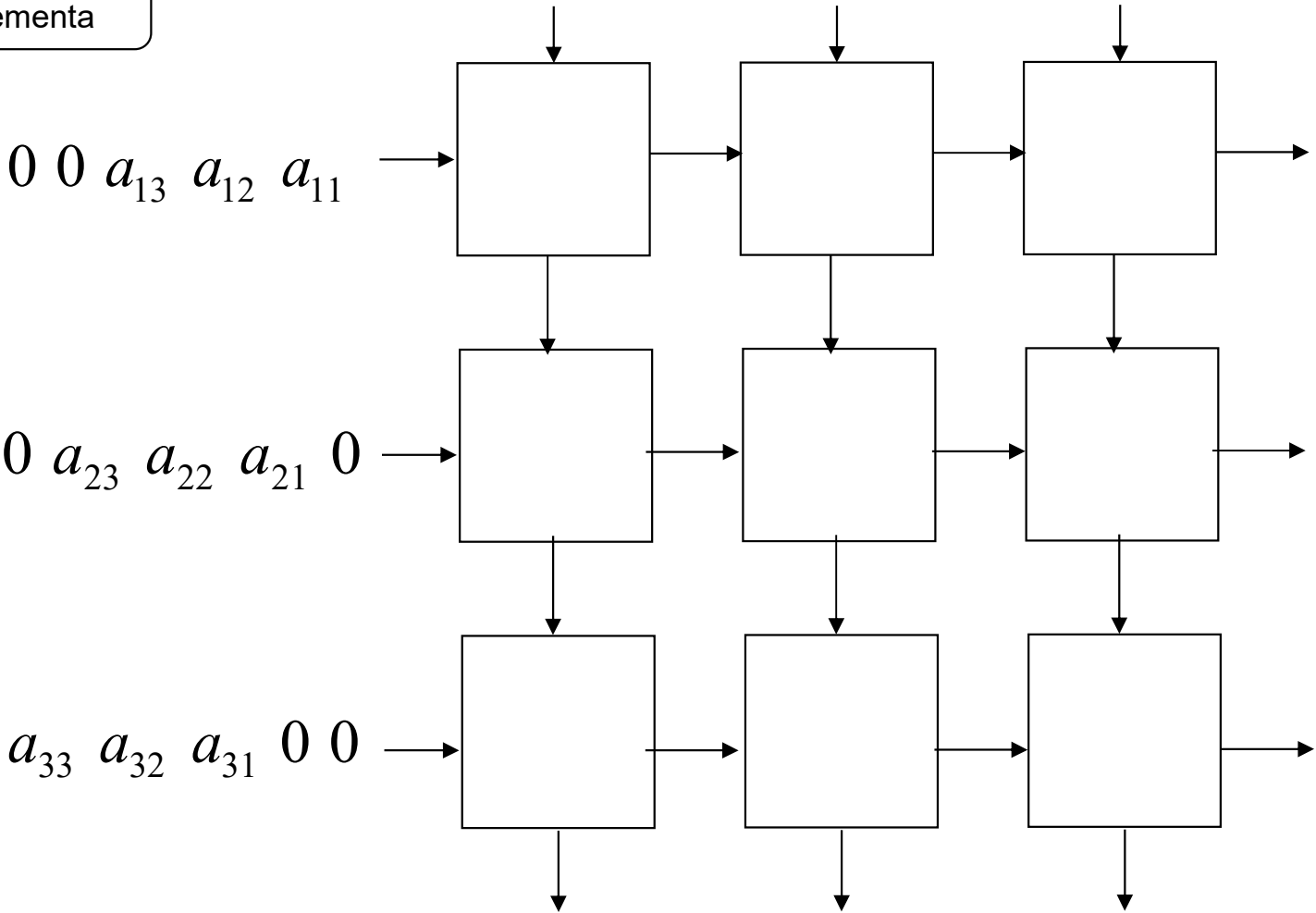


$a_{out} := a_{in}$
 $b_{out} := b_{in}$
 $c := c + a_{in} * b_{in}$

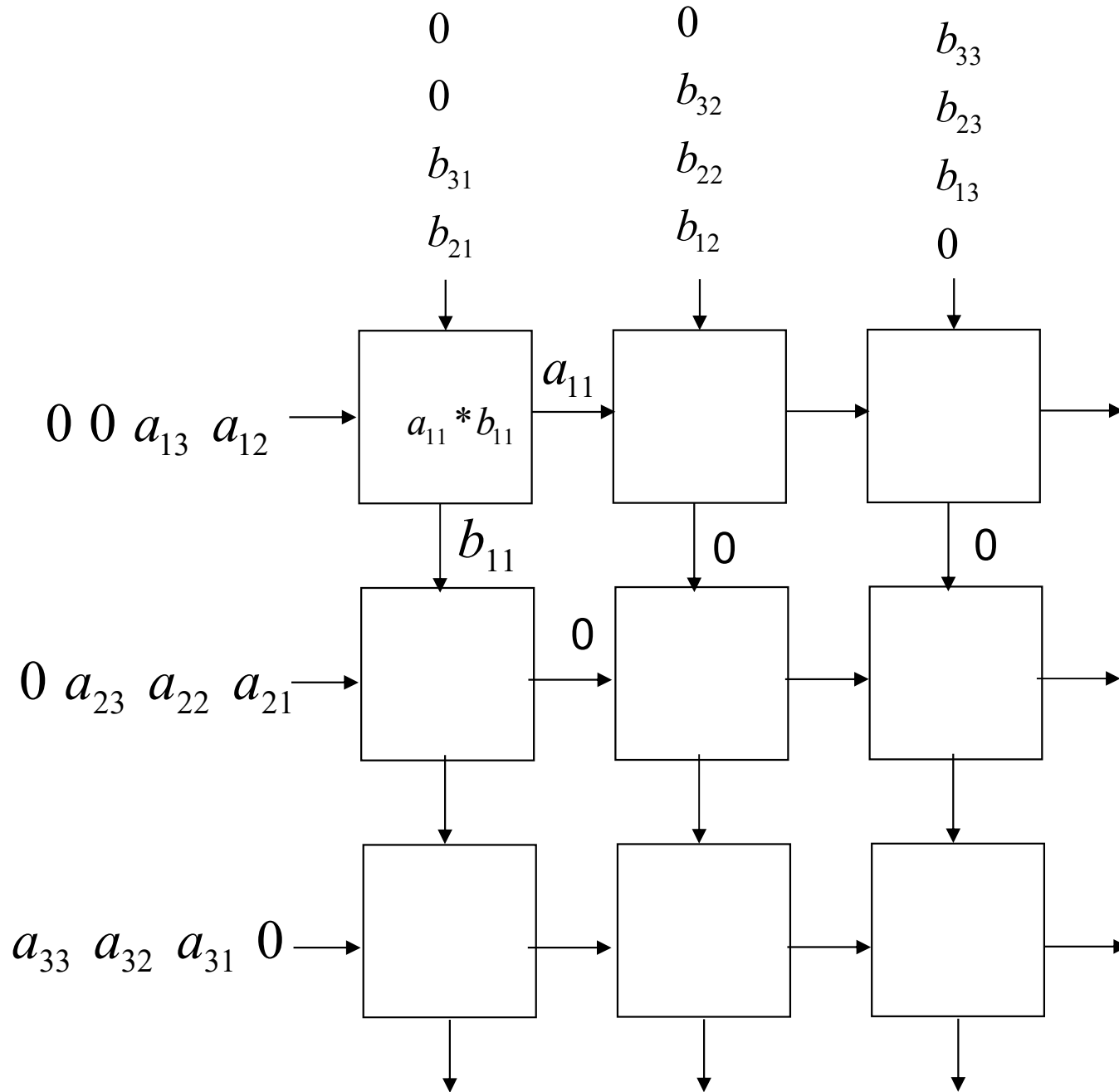
Funkcionalno svostvo
procesnog elementa

Inicijalni raspored elemenata (T=0)

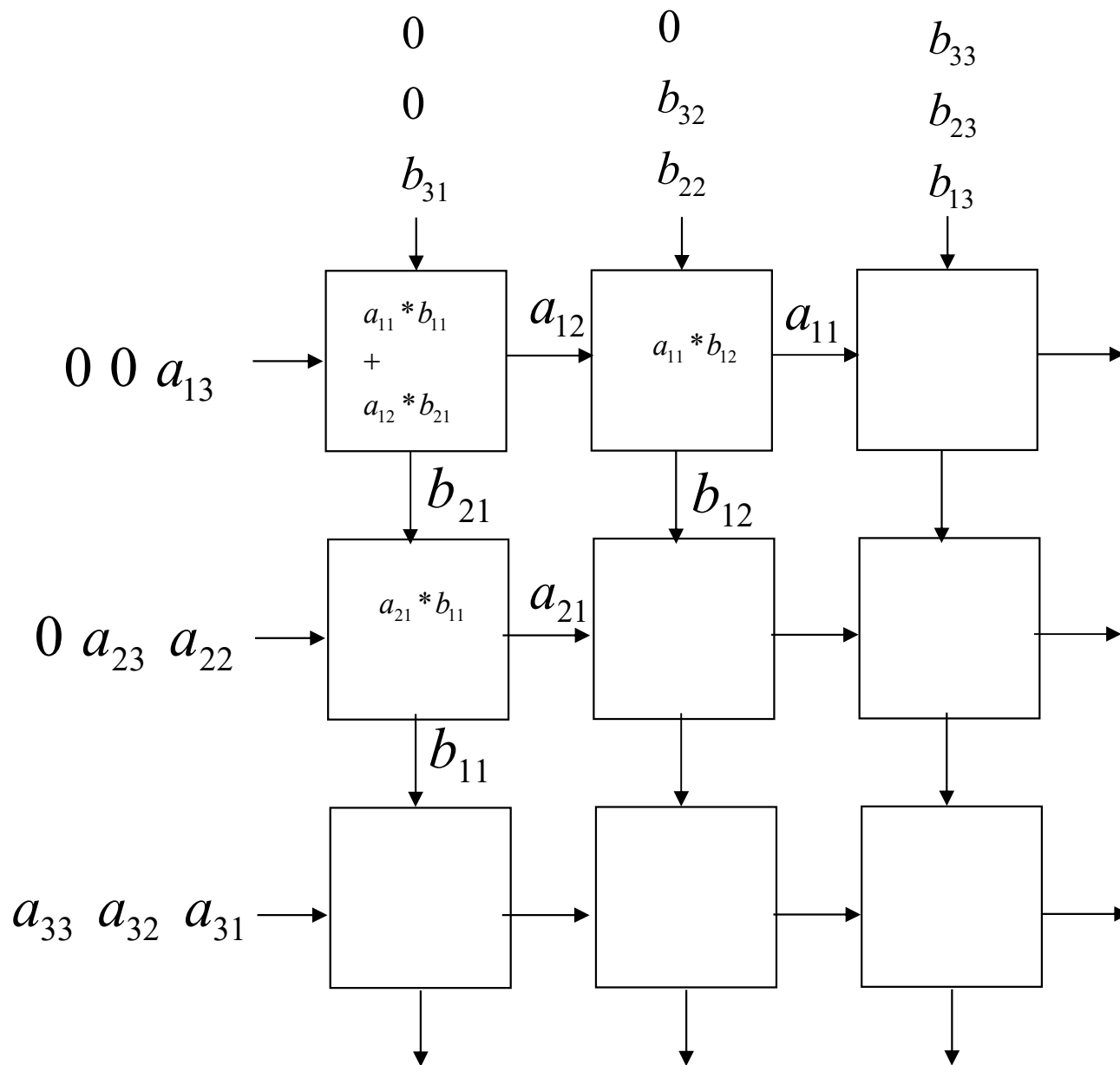
0	0	b_{33}
0	b_{32}	b_{23}
b_{31}	b_{22}	b_{13}
b_{21}	b_{12}	0
b_{11}	0	0



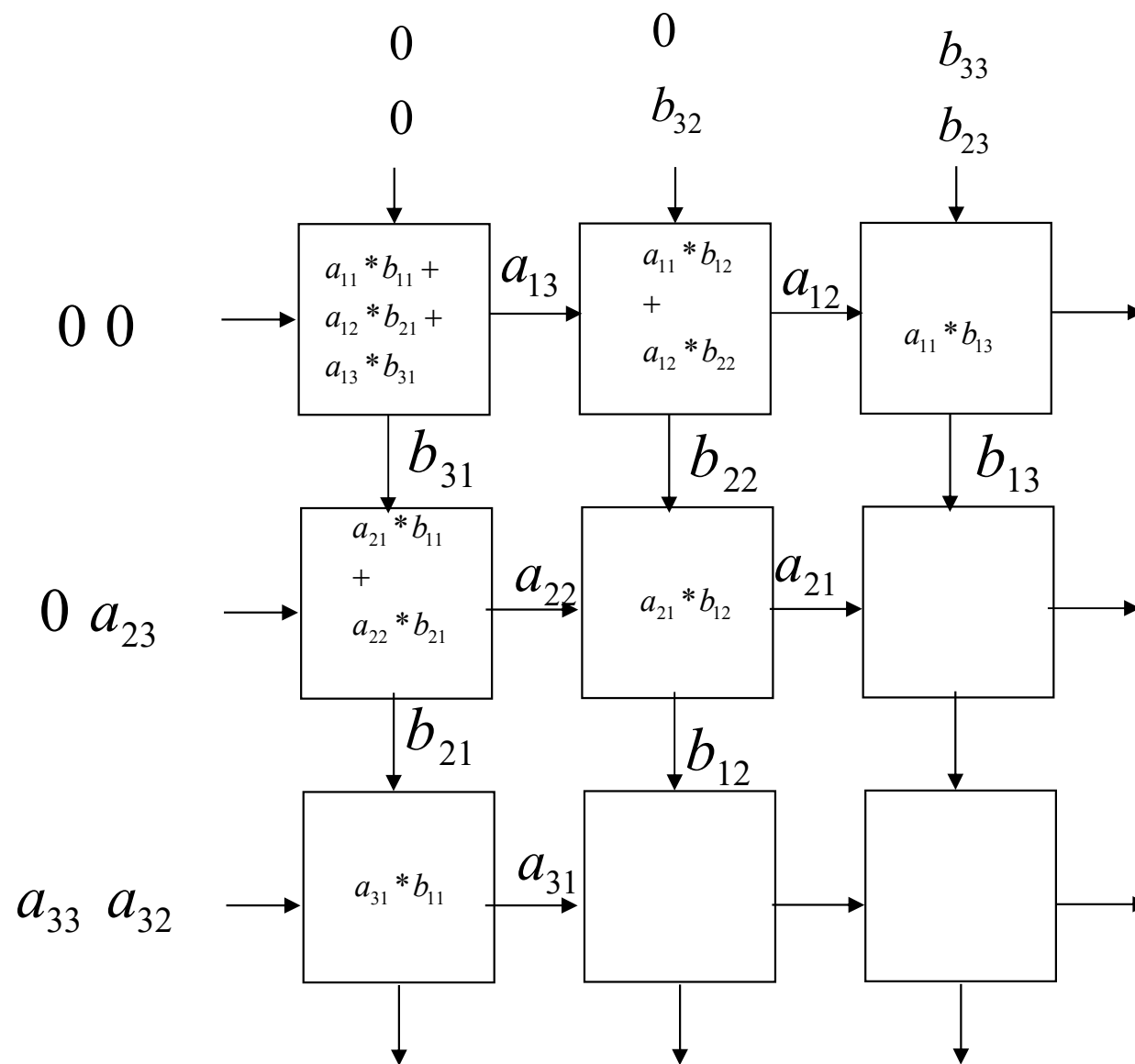
Stanje nakon prvog koraka



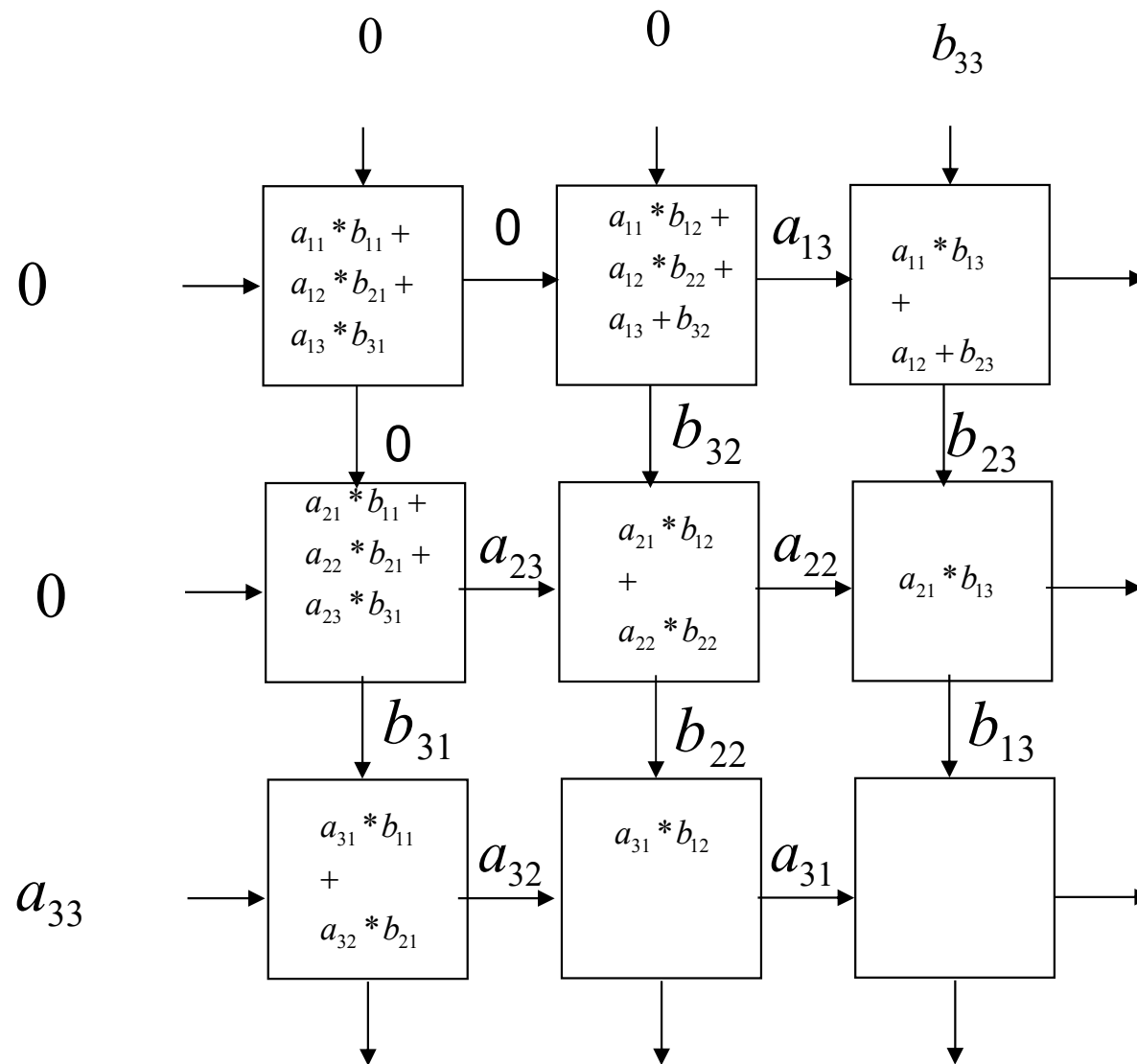
Stanje nakon drugog koraka



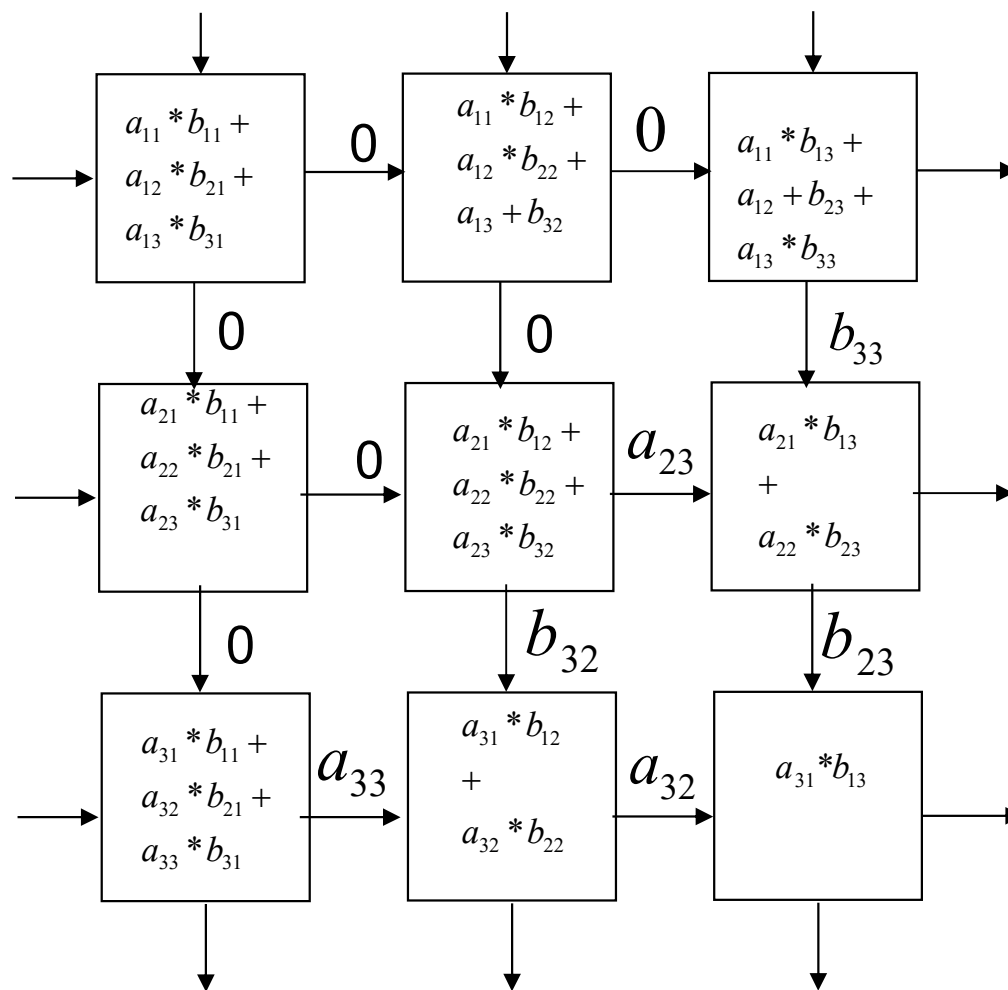
Stanje nakon trećeg koraka



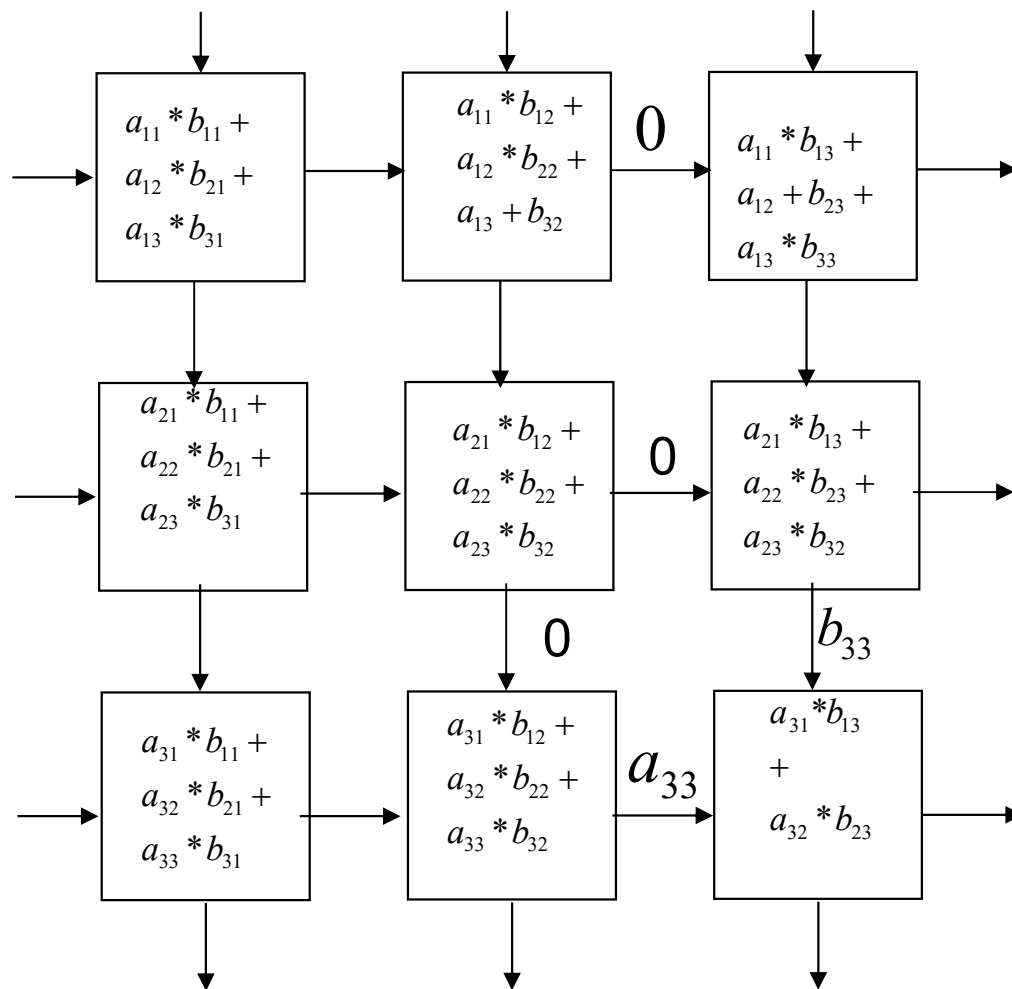
Stanje nakon četvrtog koraka



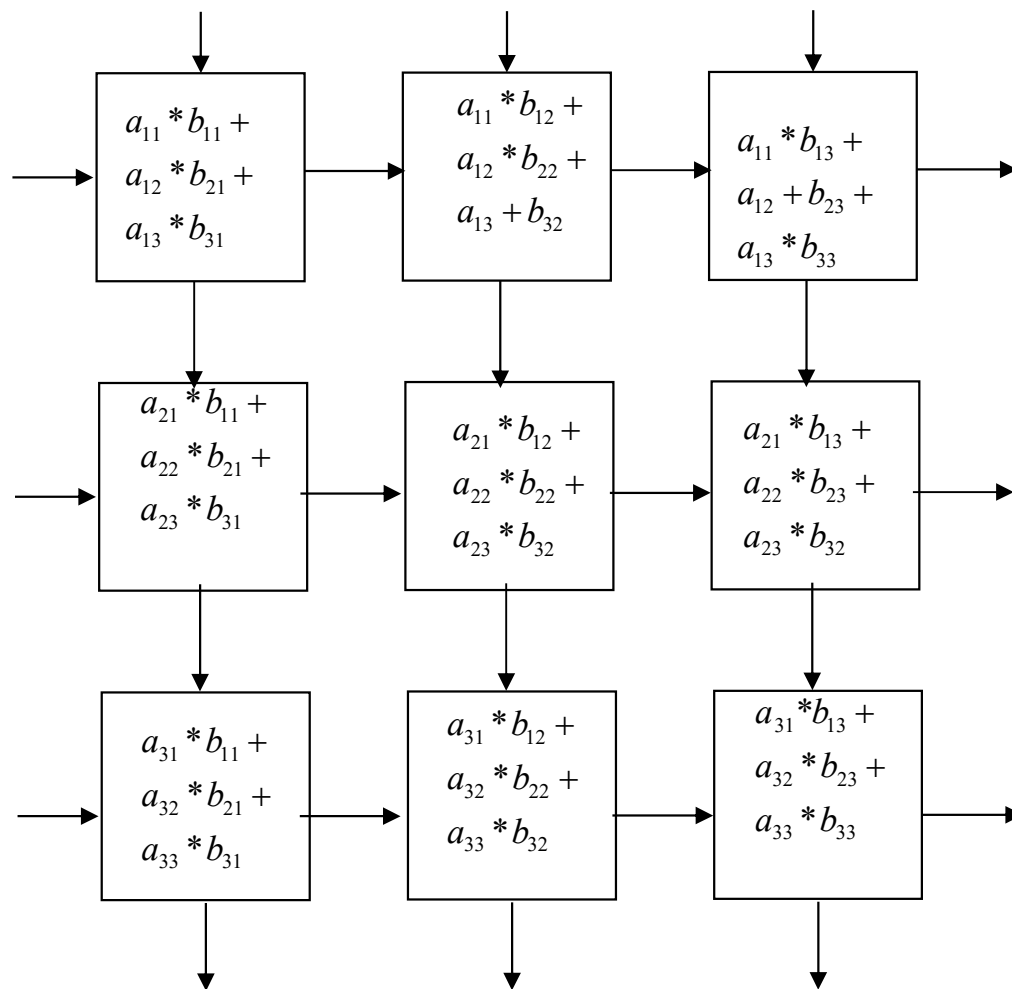
Stanje nakon petog koraka



Stanje nakon šestog koraka



Stanje nakon sedmog koraka (kraj izračunavanja)



Performanse:

Proizvod se dobija nakon $3n-2$ koraka

Na jednoprocesorskom sistemu proizvod se dobija nakon n^3 koraka

Ubrzanje: $S = \frac{T(1)}{T_{parallel}} = \frac{n^3}{3n-2} = O(n^2)$

