Paralelni sistemi

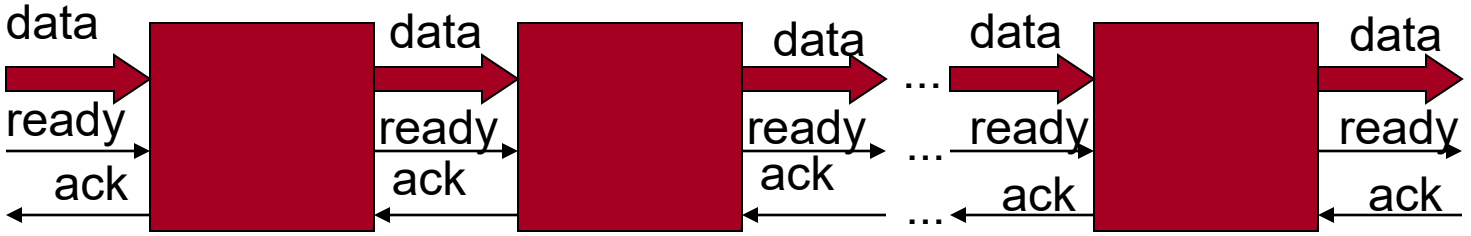
Protočnost (pipelining)

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 f1, f2, ..., fk, tako da budu zadovoljeni sledeći kriterijumi:
  - Izračunavanje osnovne funkcije f je ekvivalentno sekvencijalnom izračunavanju podfunkcija f1, f2, ..., fk.
  - izlazi prethodne podfunkcije predstavaljaju ulaze za sledeću podfunkciju u nizu podfunkcija koje se izvršavaju
  - Osim razmene podataka izmedju podfunkcija ne postoji nikakva druga zavisnost
  - Može se projektovati hardver za izračunavanje svake podfunkcije
  - Vremena potrebna ovim hardverskim jedinicama da obave individalana izračunavanja su približno jednaka

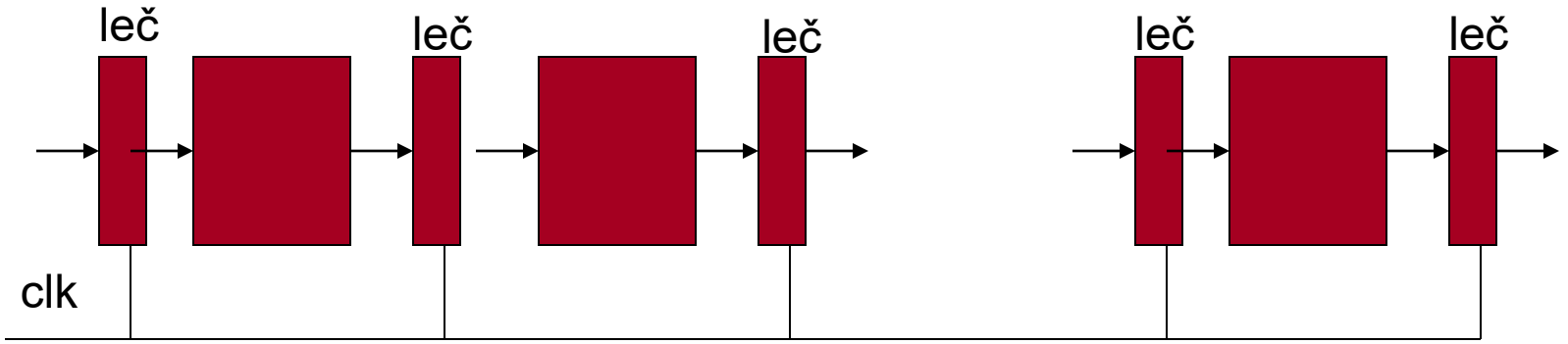
Protočnost (nast.)

- 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.

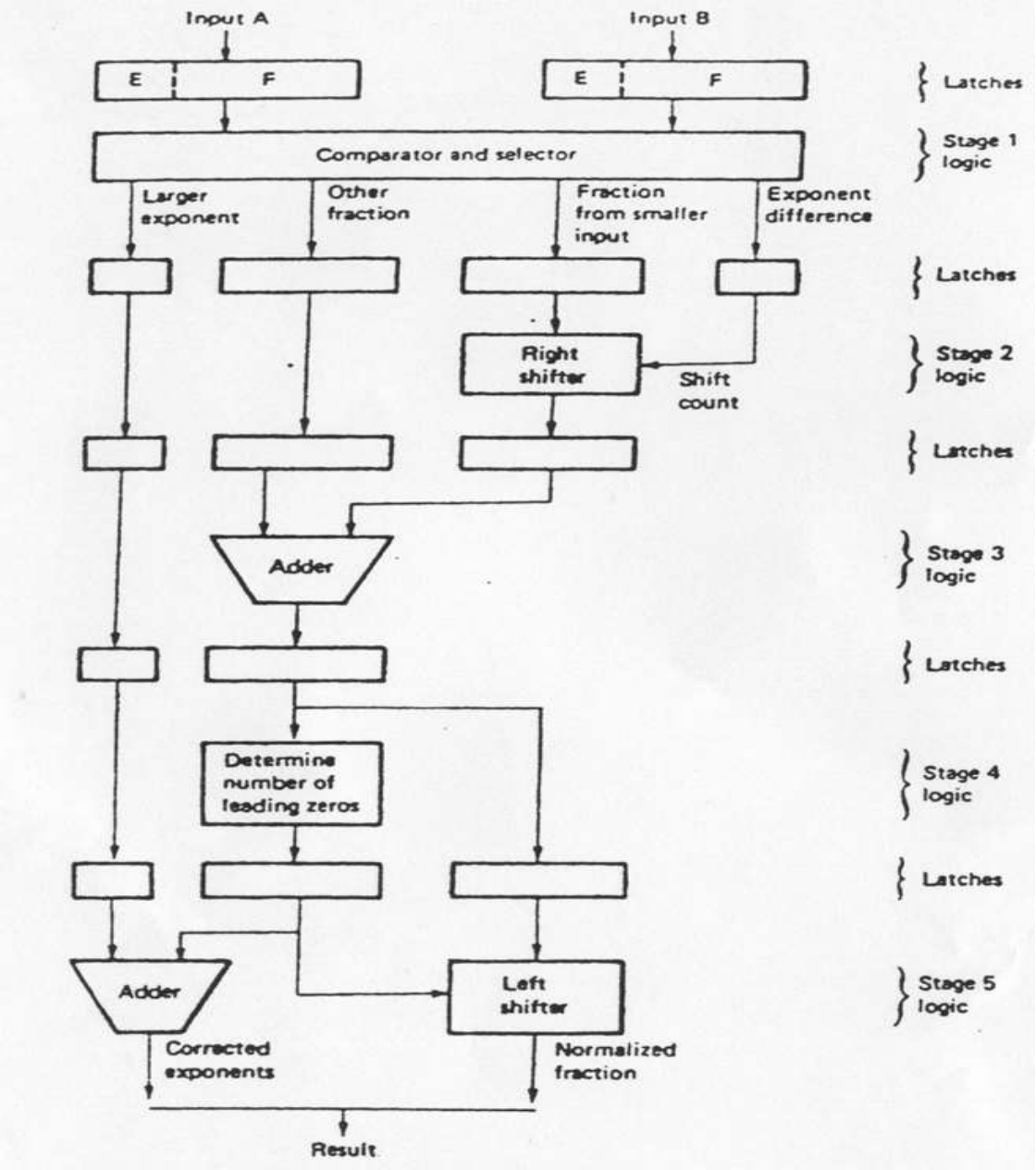
Stepen i unosi kašnjenje Ti. Klok perioda protočnog sistema iznosi

- $T=\max\{T1, T2,\ldots, Tk\}+T_L$, gde je T_L kašnjenje koje unosi leč

Primer: projektovanje protočnog FP sabiarača

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

Protočni sabirač



Protočni sabirač

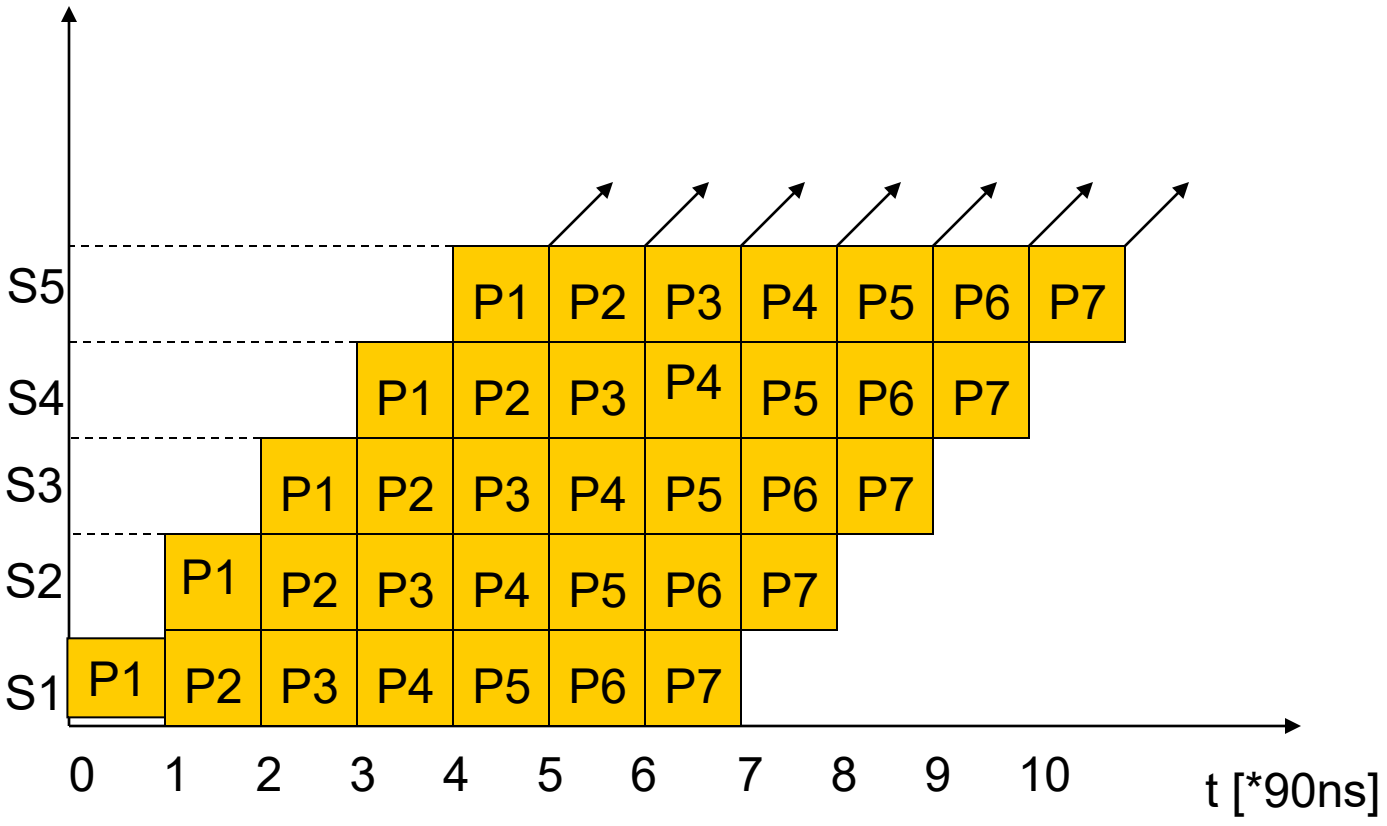
- Neaka su kašnjenja koja unose pojedini stepeni
  - T1=60 ns
  - T2=50 ns
  - T3=80 ns
  - T4=50 ns
  - T5=80 ns
  - T_L=10 ns
- Klok perioda protočnog sistema je T=max{60, 50, 80, 50, 50}+10=90ns
- Vreme potrbno neprotočnom sabiraču da sabere dva FP broja iznosi Tnp=60+50+80+50+80=320ns
- Vreme potrebno protočnom sabiraču da sabere dva FP broja iznosi Tpr=5*90=450ns

Gde je dobit od uvodjenja protočnosti?

- Ako je potrebno sabrati n parova brojeva neprotočnom sabiraču će biti potrebno
  - Tnp=n*320 ns
- a protočnom
  - Tpr=450+(n-1)*90 ns
- Za n=10
  - Tnp=10*320=3200 ns
  - Tpr=450+9*90=1360 ns
- Što je veće n performanse protočnog sistema su bolje. Za dovoljno veliko n ubrzanje protočnog sistema jednako je broju stepena, k.

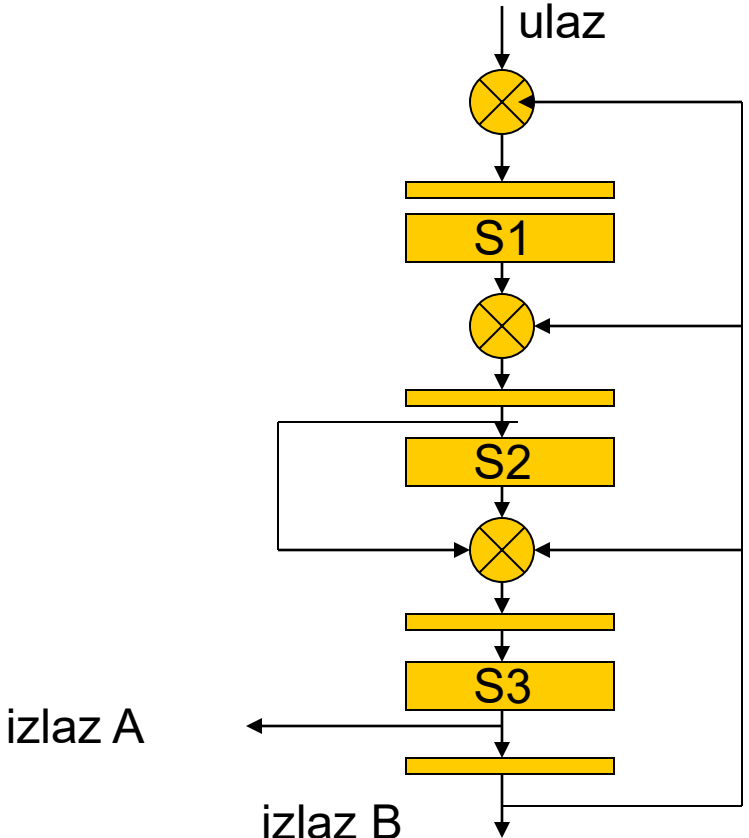
Gantov dijagram

- 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)

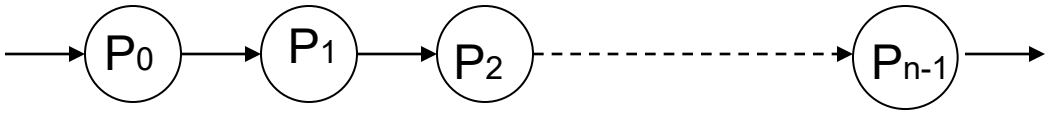


Klasifikacija protočnih sistema

- 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)

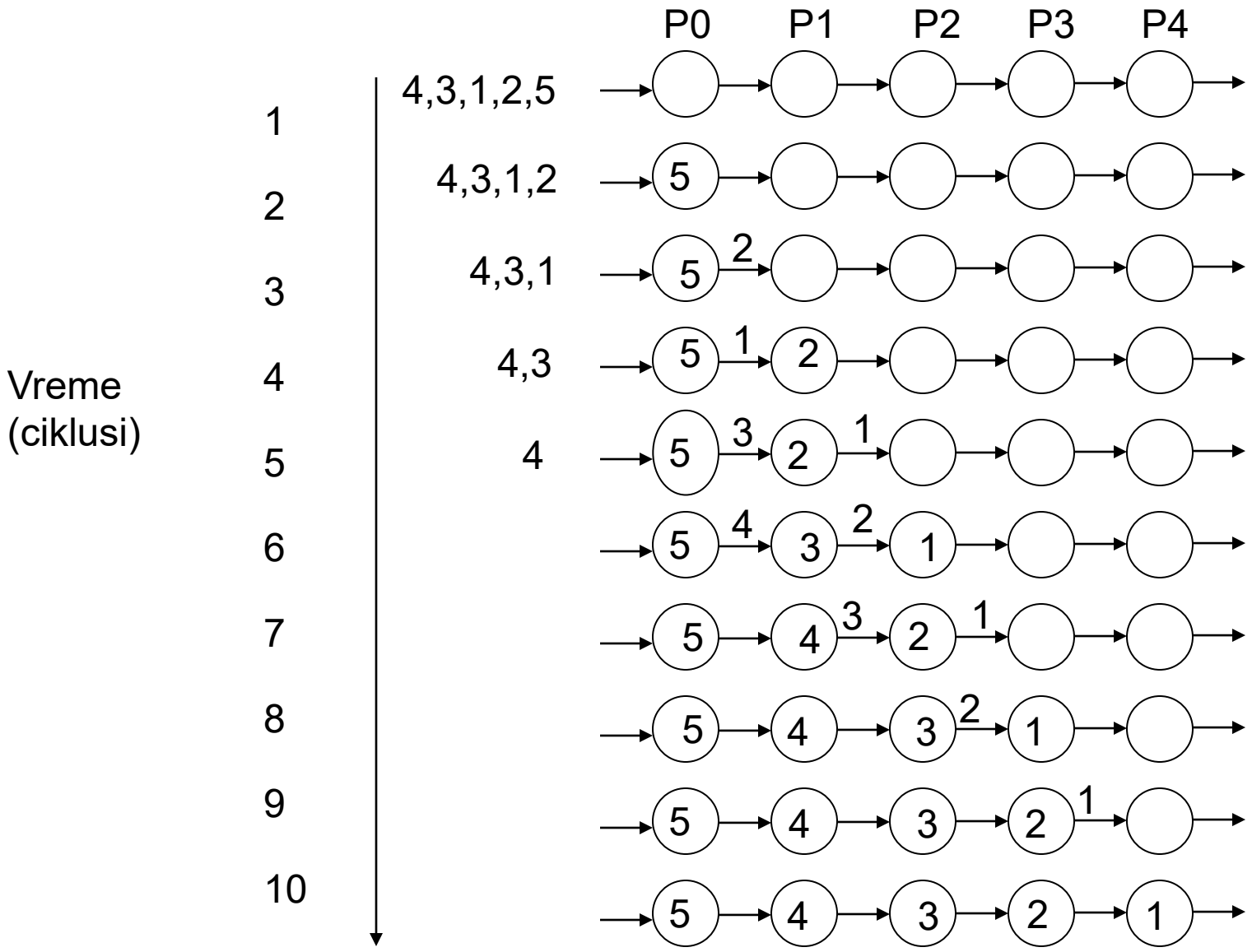
Primer – sortiranje niza

- 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 P0 prihvata niz brojeva (po jedan u datom klok ciklusu
    - Pamti najveći do tog trenutka primljeni broj
    - Prosledjuje desnom susedu manji broj
    - Svaki procesni element Pi u nizu obavlja istu operaciju (pamti njveći primljeni broj a prosledjuje manji ka Pi+1
    - Na kraju rada P0 će imati najveći broj, P1 sledeći najveći,... Pn-1 će imati najmanji broj
  - Algoritam je paralelna verzija insertion sort algoritma

Primer n=5


P0
P1
P2
P3
P4
Vreme
(ciklusi)
1
2
3
4
5
6
7
8
9
10
4,3,1,2,5
4,3,1,2
4,3,1
4,3
4
5
2
5
1
2
5
3
2
1
5
4
3
2
1
5
4
3
1
5
4
3
2
1
5
4
3
2
1
5
4
3
2
1

Primer (nast.)

- osnovni program za procesor Pi

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

- Ako ima n brojeva i n procesora, i-ti procesor (i=0,1,...,n-1) treba da primi n-i brojeva, a da prosledi n-i-1, jer se jedan od brojeva zadržava.
- Kompletan program za Pi

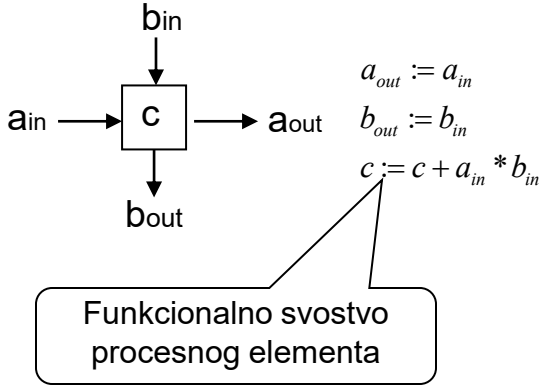
```
Broj_desnih_procesora=n-i-1;
Primi(x, Pi-1);
For (j=0: j < Broj_desnih_procesora; j++) {
primi(broj, Pi-1)
If (broj > x) {posalji(x,Pi+1);
x=broj;}
else posalji(broj, Pi+1);
}
```

Analiza performansi

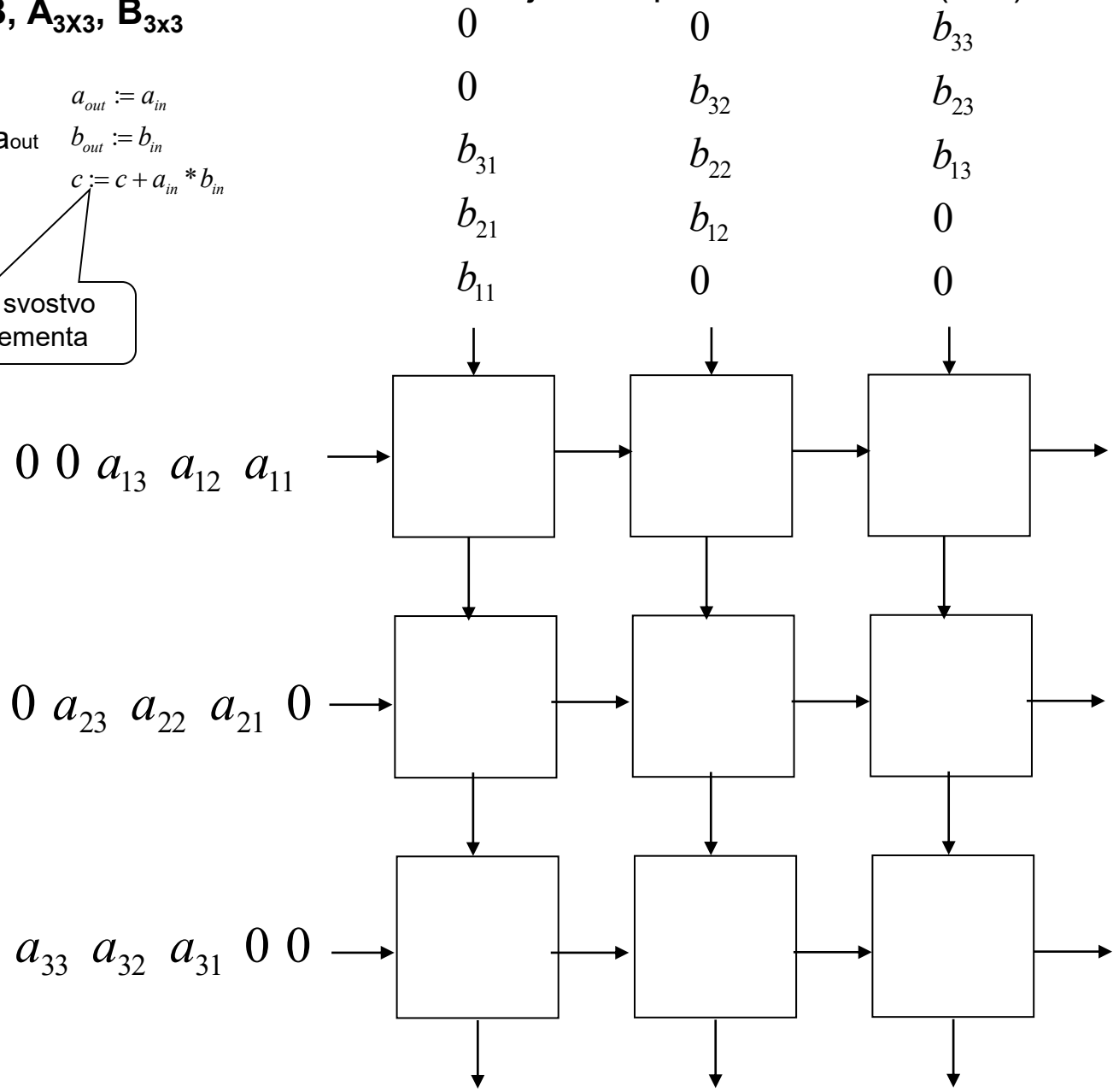
- Sekvencijalni algoritam
  - Usvojimo da poredjenje i dodeljivanje vrednosti traju 1 vremensku jedinicu;
  - Sekvencijalno vreme izvršenja
    - $T_1=(n-1)+(n-2)+...+2+1=n(n-2)/2 \cong O(n^2/2)$
- Paralelni algoritam
  - Paralelna implementacija ima n+n-1=2n-1 ciklusa, ako ima n procesora i n brojeva koje treba sortirati
    - Svaki procesor u u jednom ciklusu obavlja poredjenje i prijem i predaju podataka (izuzev poslednjeg)
      - Tciklusa = 1 + tkomunikacije
    - Potrebno je još n cikusa da se protočni sistem isprazni
    - Ukupno vreme rada protočnog sistema iznosi
    - $T_p=(3n-1)$ Tciklusa
  - Ubrzanje

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

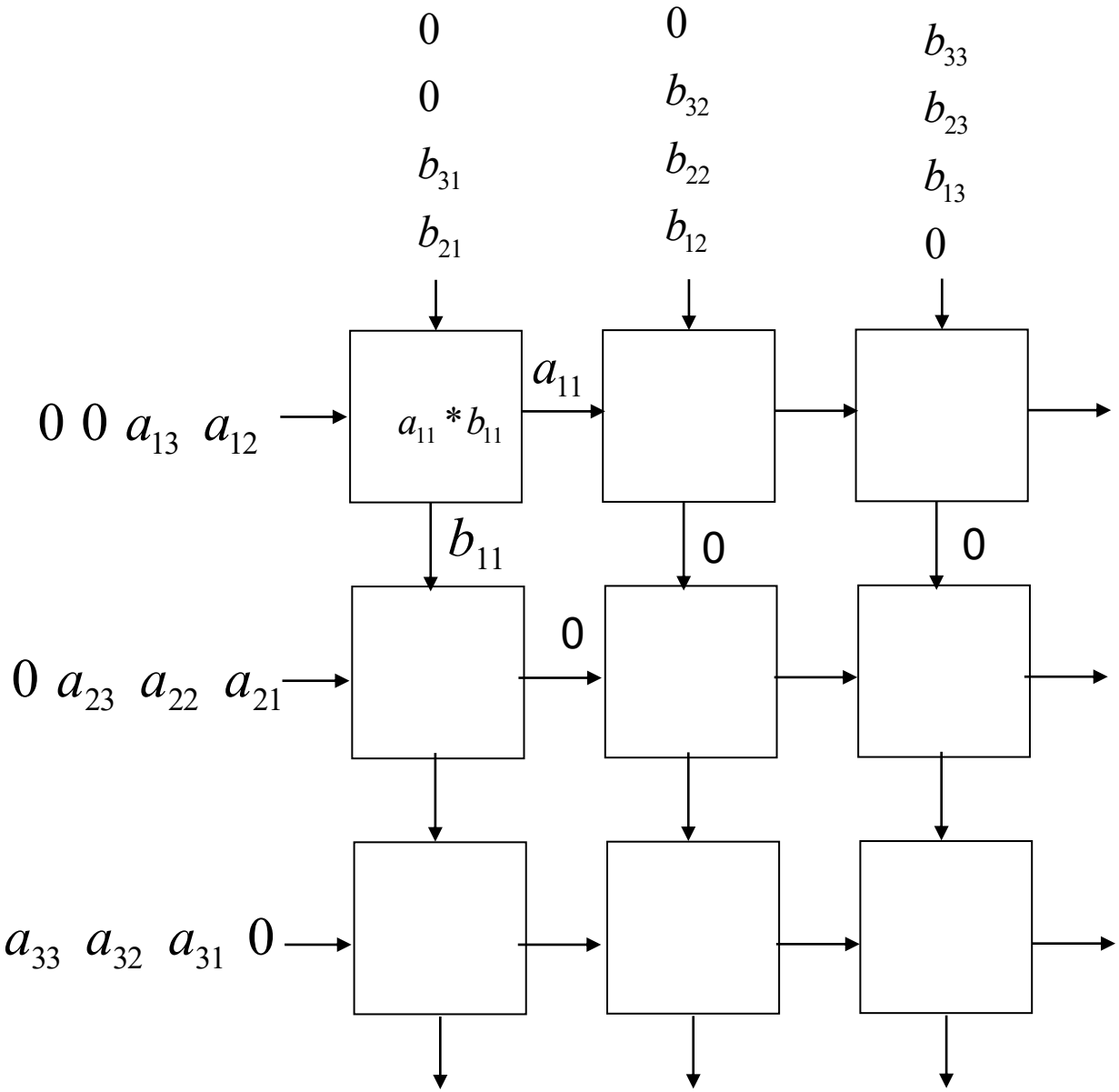
Primer: Množenje
Matrica C=A*B, A_{3X3}, B_{3x3}



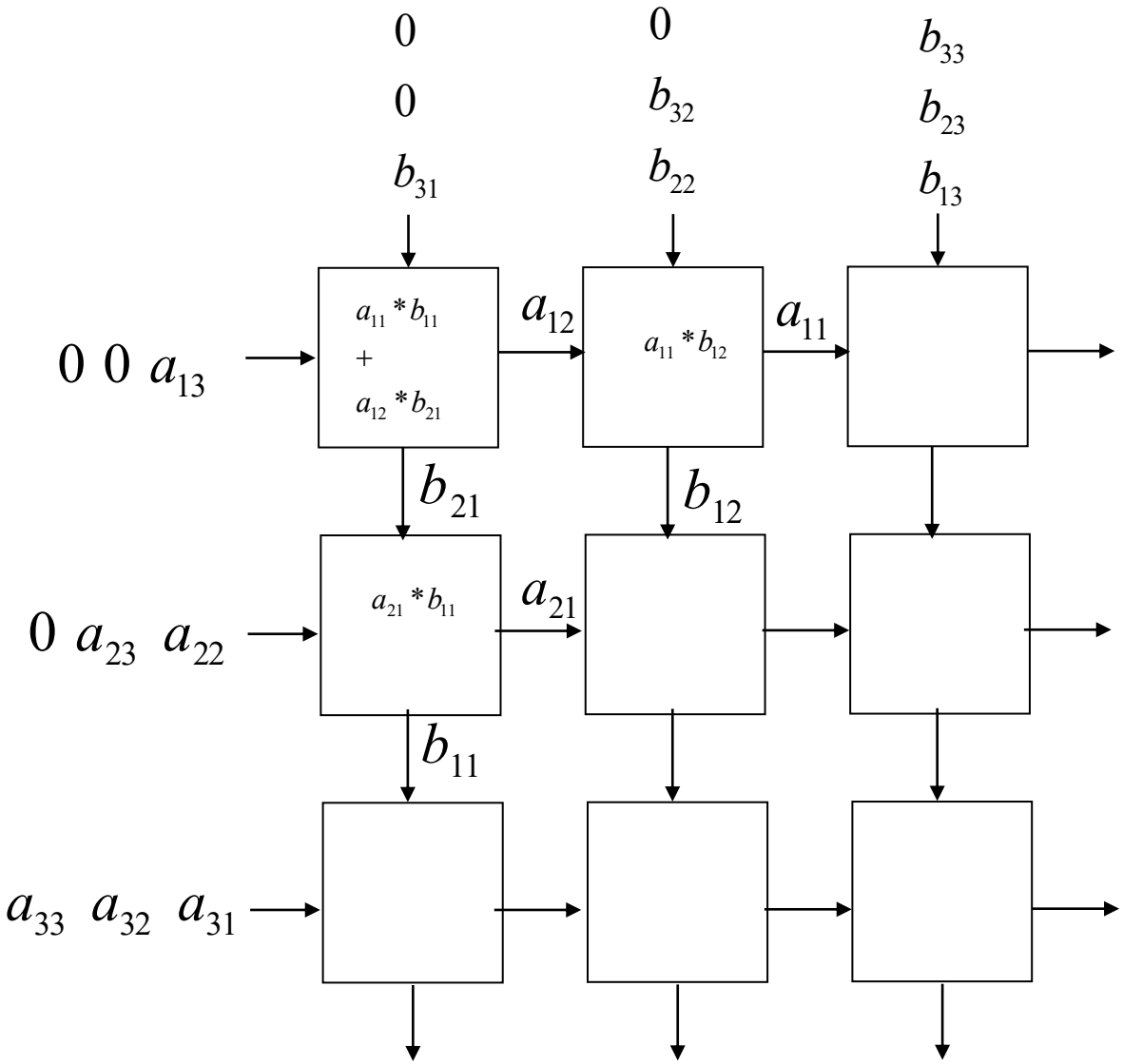
Inicijalni raspored elemenata (T=0)



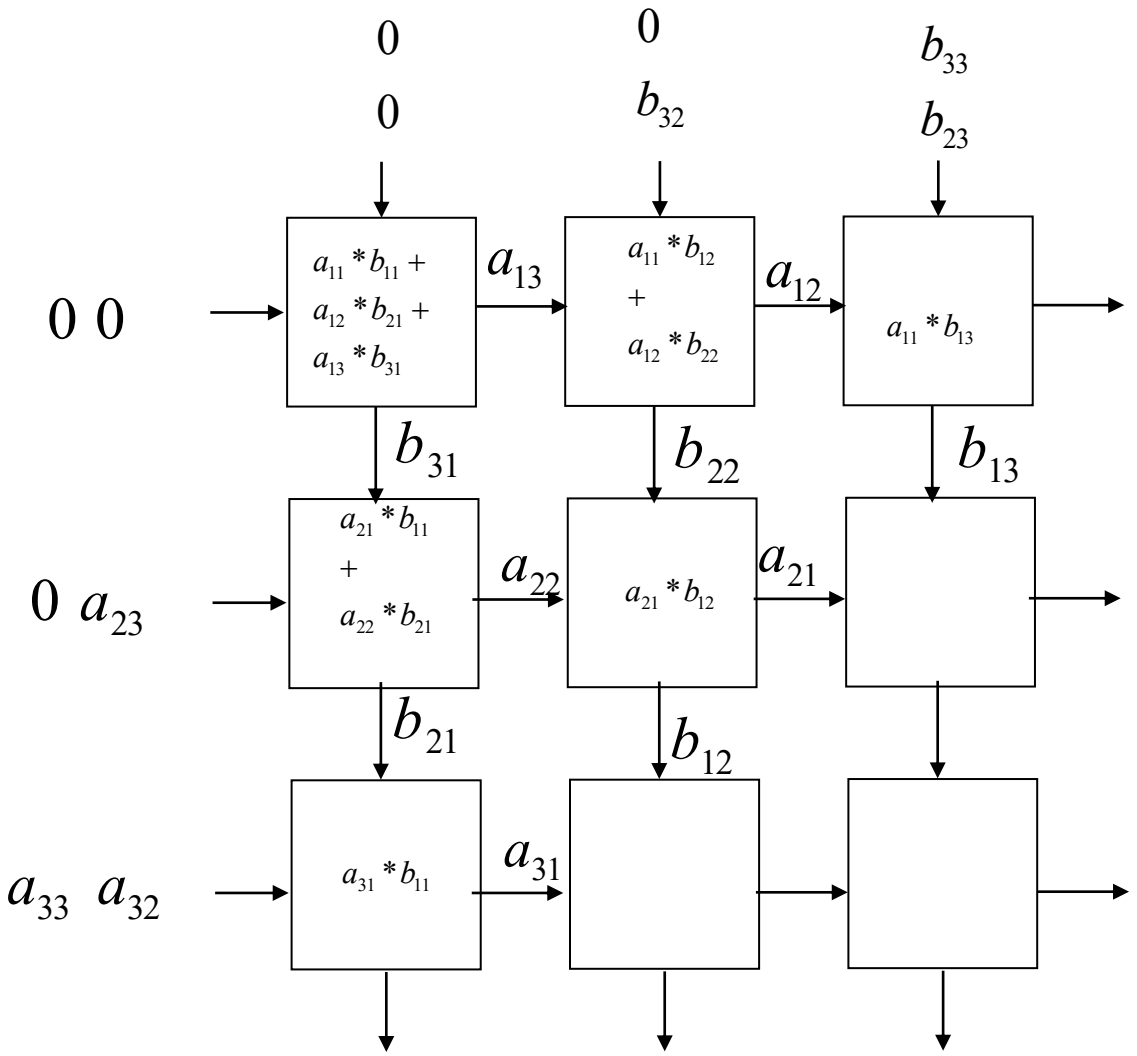
Stanje nakon prvog koraka



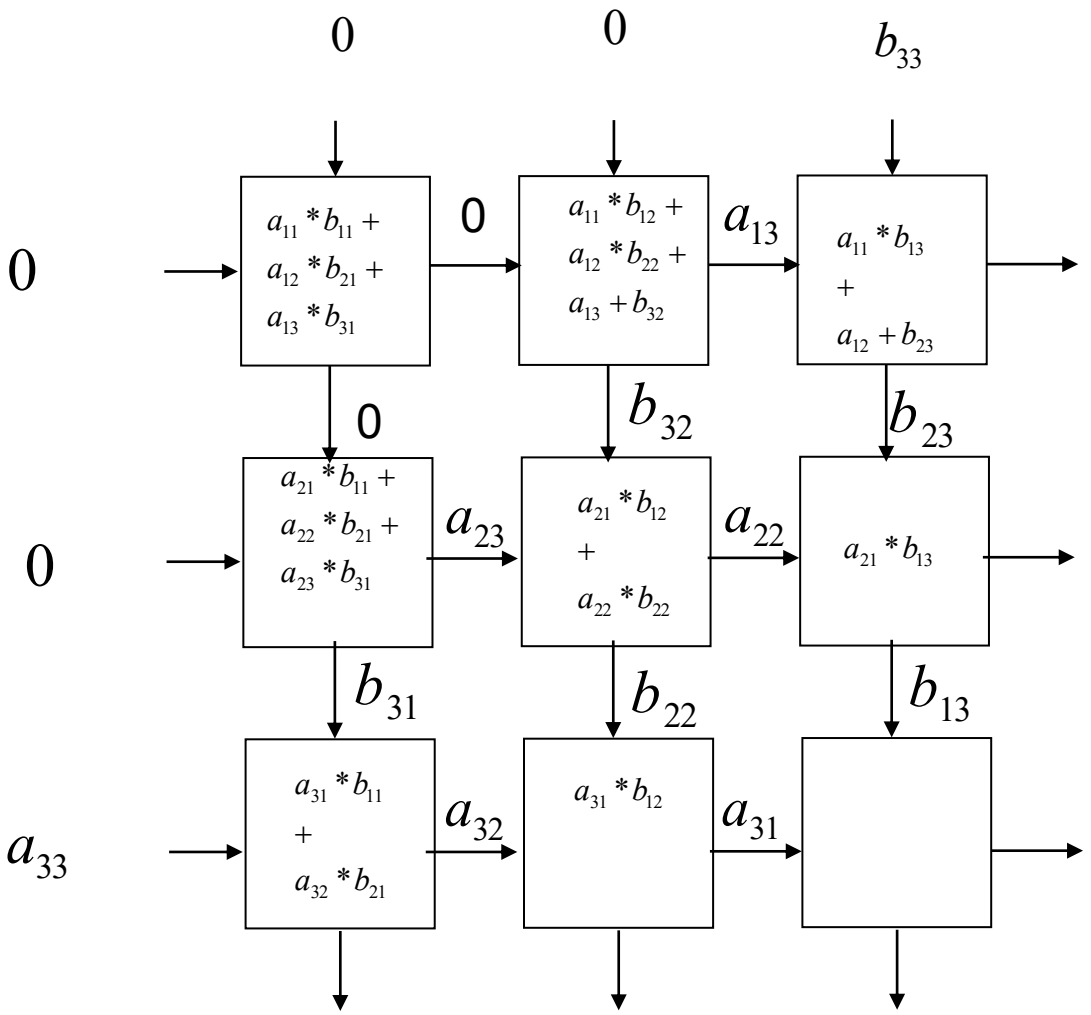
Stanje nakon drugog koraka



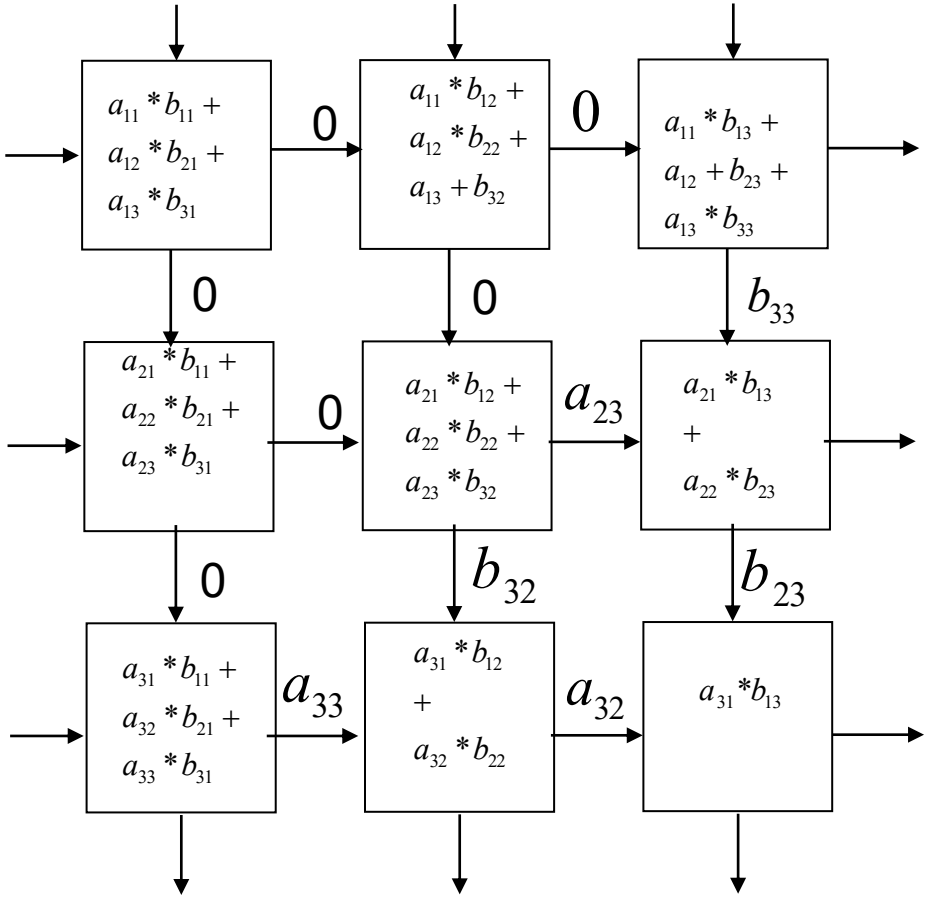
Stanje nakon trećeg koraka



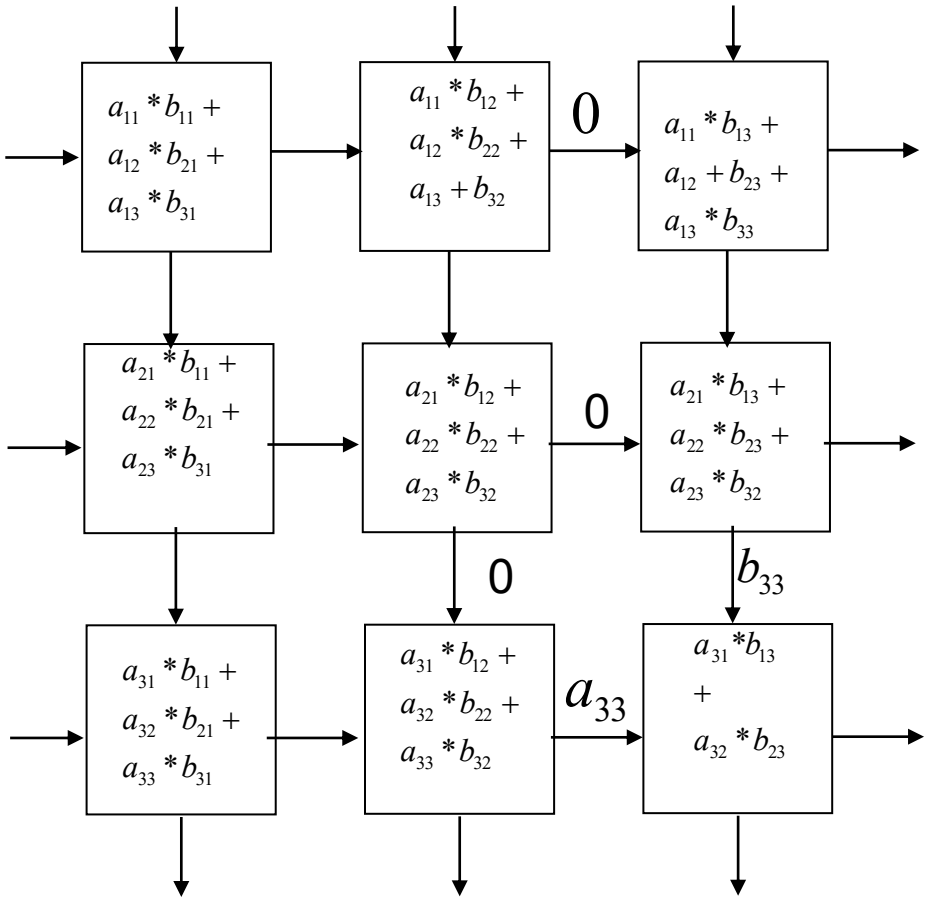
Stanje nakon četvrtog koraka



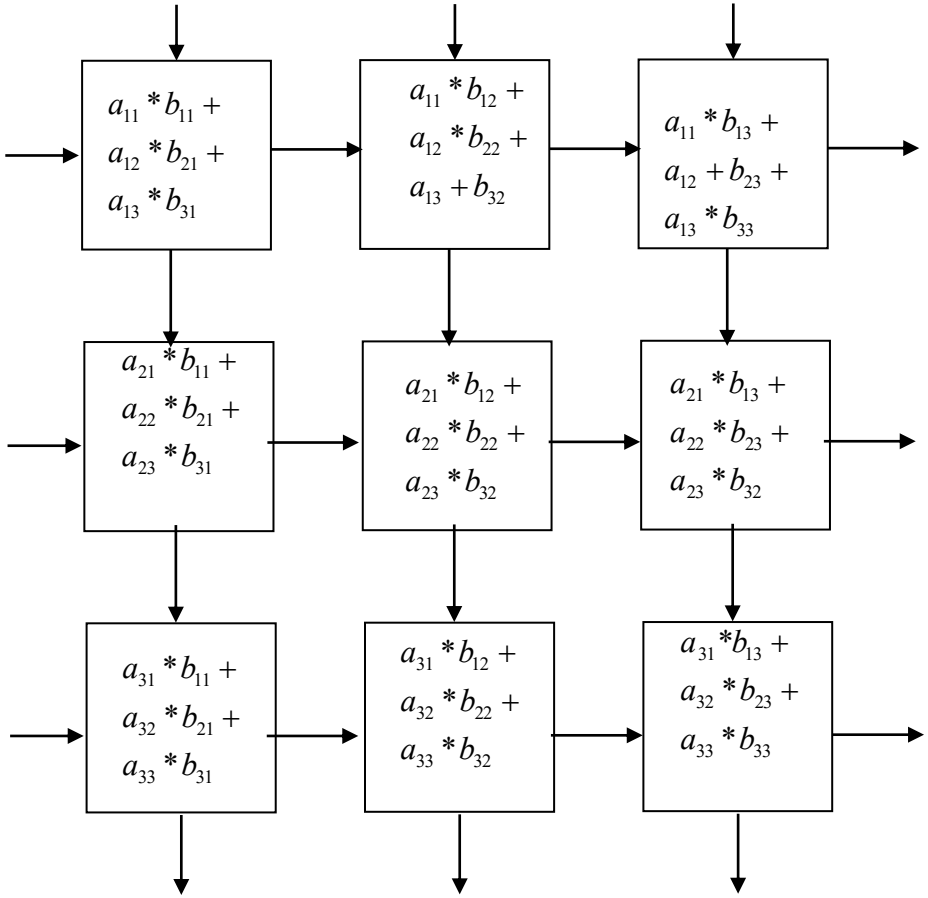
Stanje nakon petog koraka



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