# Optimizacija vezja na ravni registrov

Digitalni elektronski sistemi 2024/25

Andrej Trost

UNIVERZA V LJUBLJANI
FE
Fakulteta za elektrotehniko

# Vzporedna (paralelna) izvedba nalog

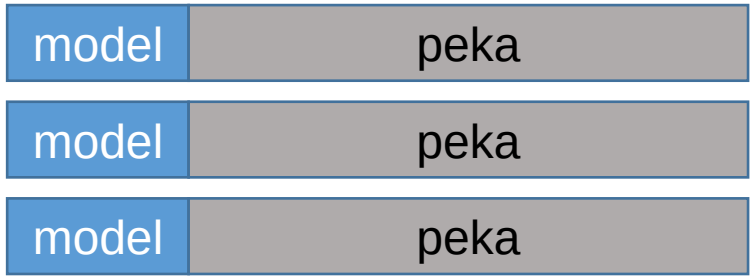
- enota: skupina vhodov, ki se obdela in pride na izhod
  - npr.  en vzorec zvoka
- zakasnitev ali latenca (latency)
  - čas za prehod enote od vhoda do izhoda
- prepustnost (throughput)
  - število obdelanih enot v enoti časa

Vzporedna izvedba nalog povečuje prepustnost!

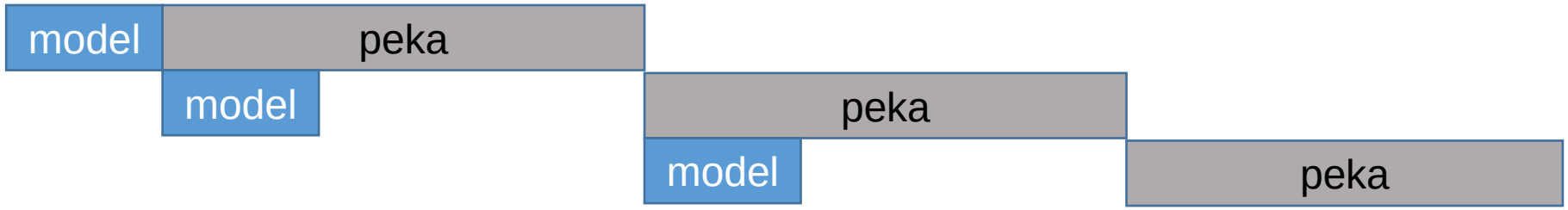
# Naloga: pečemo piškote

Koraki: 1. iztiskanje modelov in 2. pečenje v pečici


- Možnosti hitrejše izvedbe?
- 1. prostorska (več modelov, pečic in kuharjev)

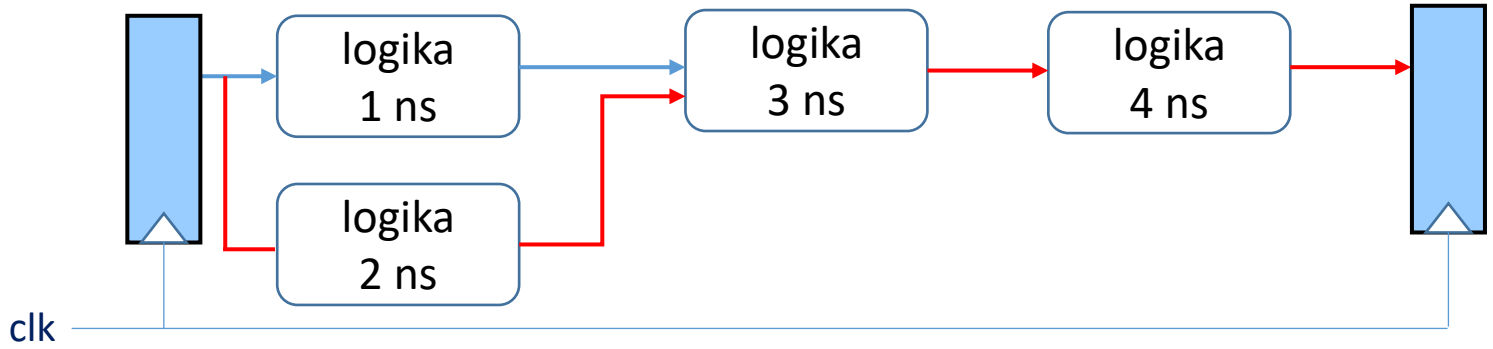



- 2. časovna (tekoči trak, cevovod)
  - nalogo razdelimo na več opravil, ki potekajo hkrati

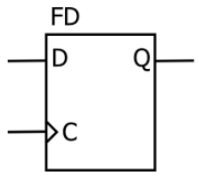


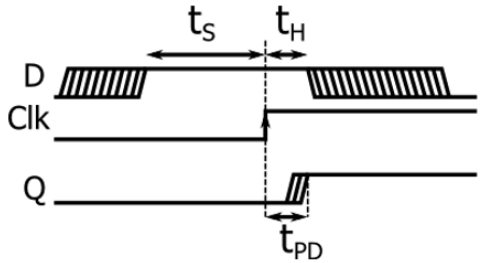
# Zakasnitve v sekvenčnem vezju

- Zmogljivost vezja omejujejo zakasnitve v kombinacijskem delu: prenos pri seštevanju,…
  - frekvenco ure omejuje povezava z največjo zakasnitvijo, ki se imenuje kritična pot



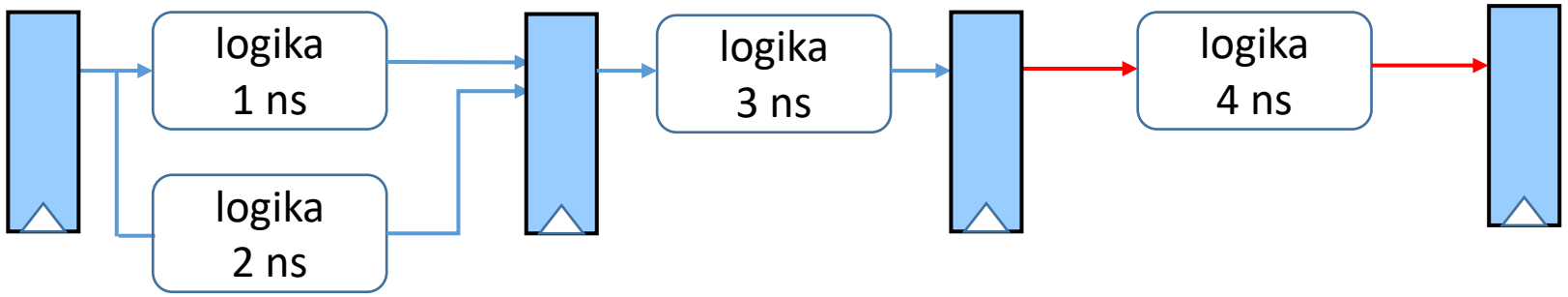

- Parametri registrov
  $t_{pd}$=0.3 ns, $t_s$ = 0.2 ns




  - celotna zakasnitev: $T_c$ = 0,3 + 2 + 3 + 4 + 0,2 = 9,5 ns

# Vezje s cevovodom (pipeline)

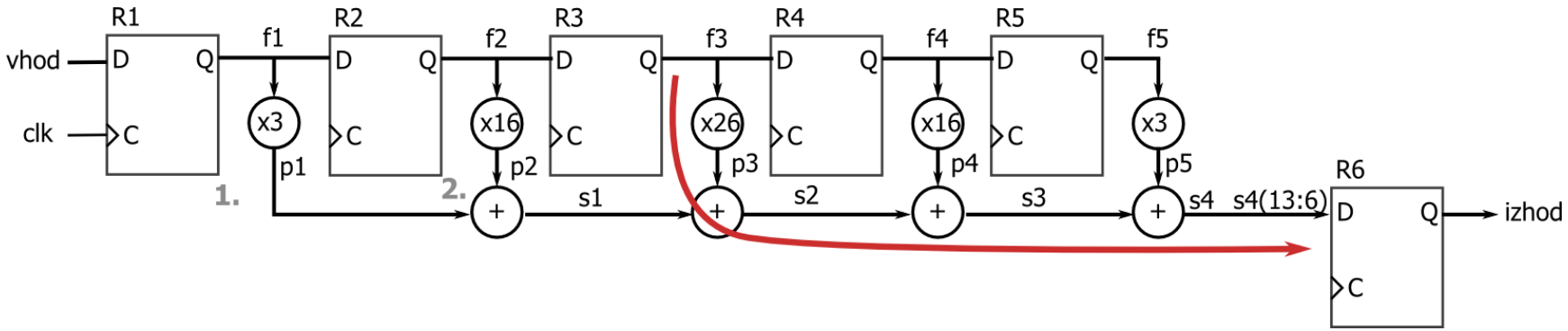



Dodamo registre med kombinacijsko logiko

- največja zakasnitev: $T_c = 0,3 + 4 + 0,2 = 4,5$ ns
- prepustnost vezja se poveča!

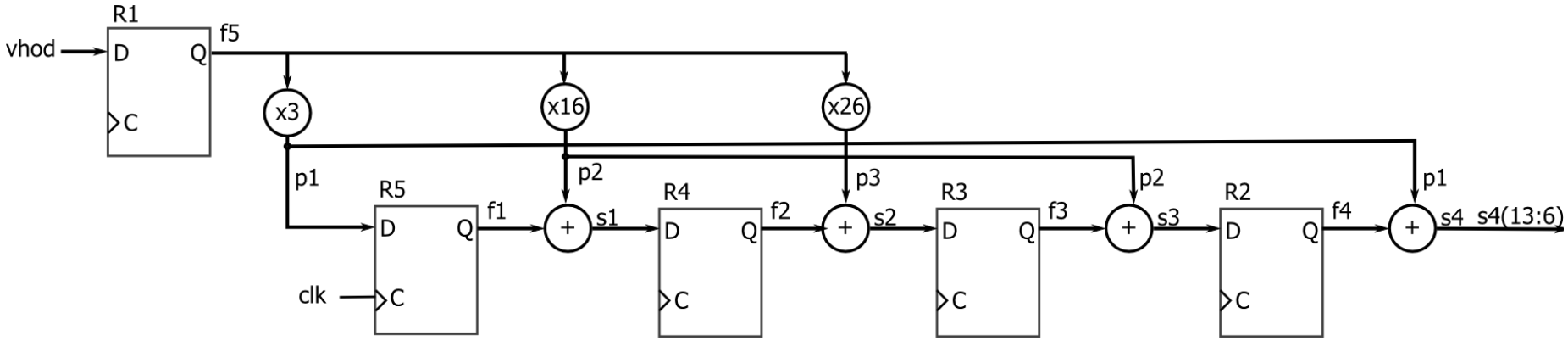
# 1. vezje: Gaussovo sito

- Obdelava zvoka s konvolucijo, cevovodno vezje
  - zakasnitve v seštevalnikih
  - množenje s 26 = 11010 se spremeni v 2 seštevanji



# Optimizirano Gaussovo sito

- Preuredimo operacije, da zmanjšamo kritično pot
  - množenje z 26 in eno seštevanje na kritični poti




- Kakšna je zakasnitev in prepustnost?
- 1 cikel ure, zaradi krajše kritične poti bo prepustnost višja

# Zakasnitve odvisne od tehnologije

ASIC: v integriranih vezjih prevladujejo zakasnitve v logiki

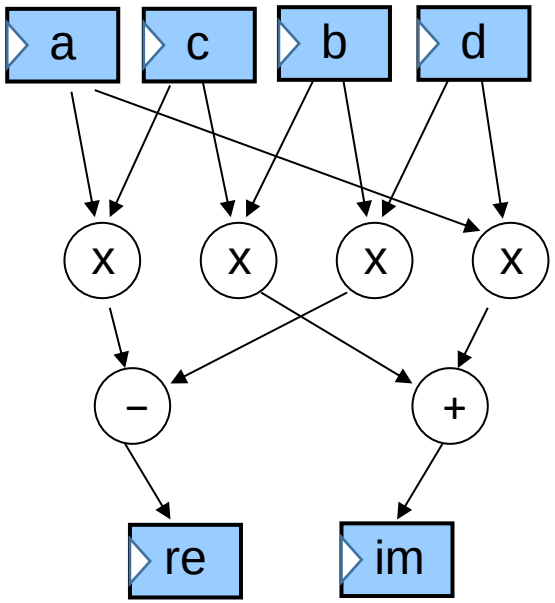
- zmanjšamo s topologijo vrat, obremenitvijo in načrtovanjem transistorjev

FPGA: programirljiva vezja imajo velike zakasnitve v povezavah

- na križiščih povezav so elektronska stikala, ki vnašajo zakasnitev pri prenosu signalov
- izbira kratkih in namenskih povezav (optimizacija v programski opremi)

# 2. vezje: kompleksno množenje

$$(a + bi)(c + di) = ac - bd + (bc + ad)i$$

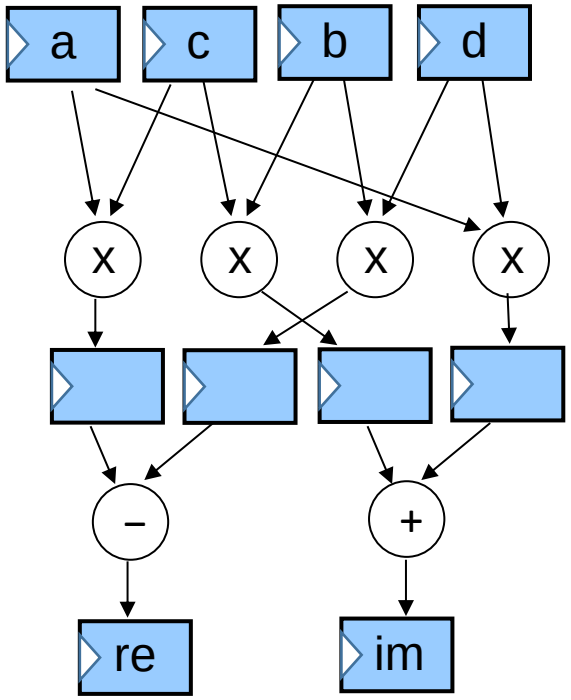



```
entity complex
 a,b,c,d: in u8;
 re,im: out u16;
 p1,p2,p3,p4: u16
begin
 p1 = a*c; p2 = b*d;
 p3 = b*c; p4 = a*d;
 re <= p1-p2
 im <= p3+p4
end
```

- 4 množilniki, seštevalnik, odštevalnik
- FPGA Cyclone IV: 1424 LE,  128 FF, 106 MHz (zakasnitev : 1 cikel)

# 2. vezje: kompleksno množenje, cevovod

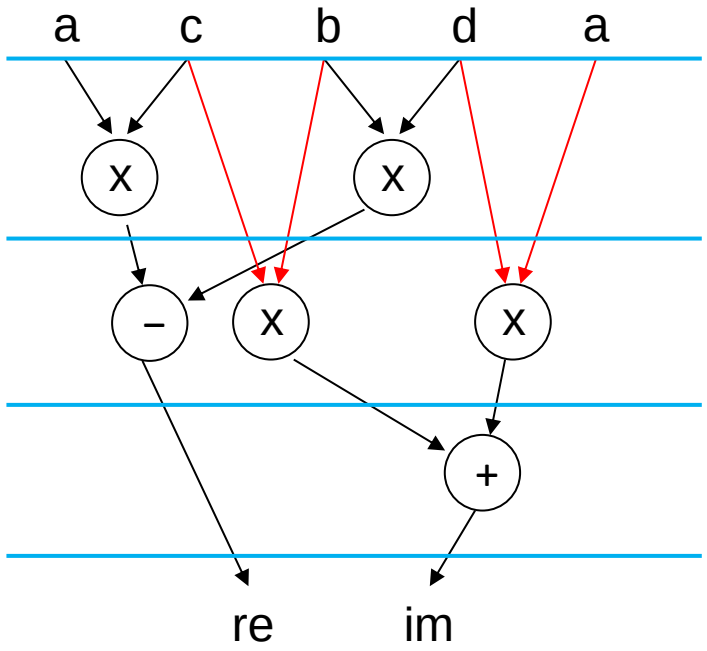
$$(a + bi)(c + di) = ac - bd + (bc + ad)i$$




- 1424 LE, 128 FF, 106 MHz (1 cikel)
- 1424 LE, 256 FF, 129 MHz (2 cikla)
- 1550 LE, 320 FF, 152 MHz (3 cikli)
  - množilnik razdeljen na 2 stopnji
  - prepustnost 152 M op/s

# Optimizacija površine vezja

deljena uporaba množilnikov in gradnika addsub




- 2 mult, 1 addsub
- 3 računski cikli

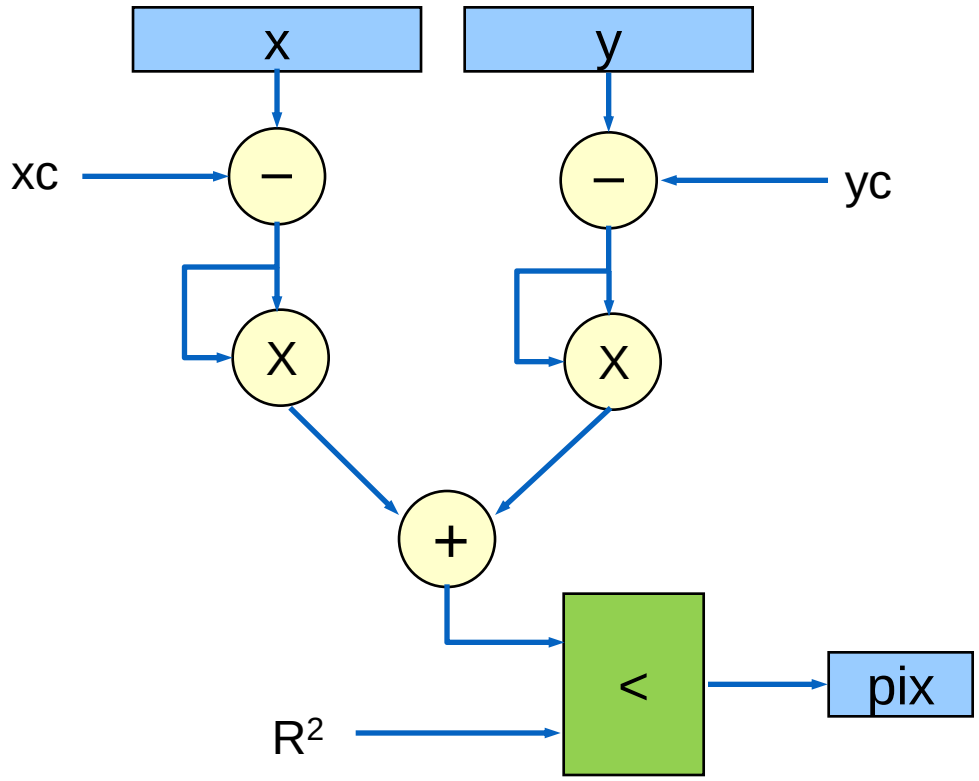
```
p1 = m1*c; p2 = m2*d
if st=1 then
  m1<=a; m2<=b;
else
  m1<=b; m2<=a;
end
if st=2 then
  re <= p1-p2
elsif st=3 then
  im <= p1+p2
end
```

- 1424 LE, 128 FF, 106 MHz (1 cikel)
- 790 LE, 98 FF, 133 MHz
  - zakasnitev: 3 cikli
  - prepustnost: 133/3 = 44 M op/s

# 3. vezje: izračun točk v krogu

Ali točka (x,y) leži znotraj krožnice (xc, yc, $R^2$)?

$(x-xc)^2 + (y-yc)^2 < R^2$



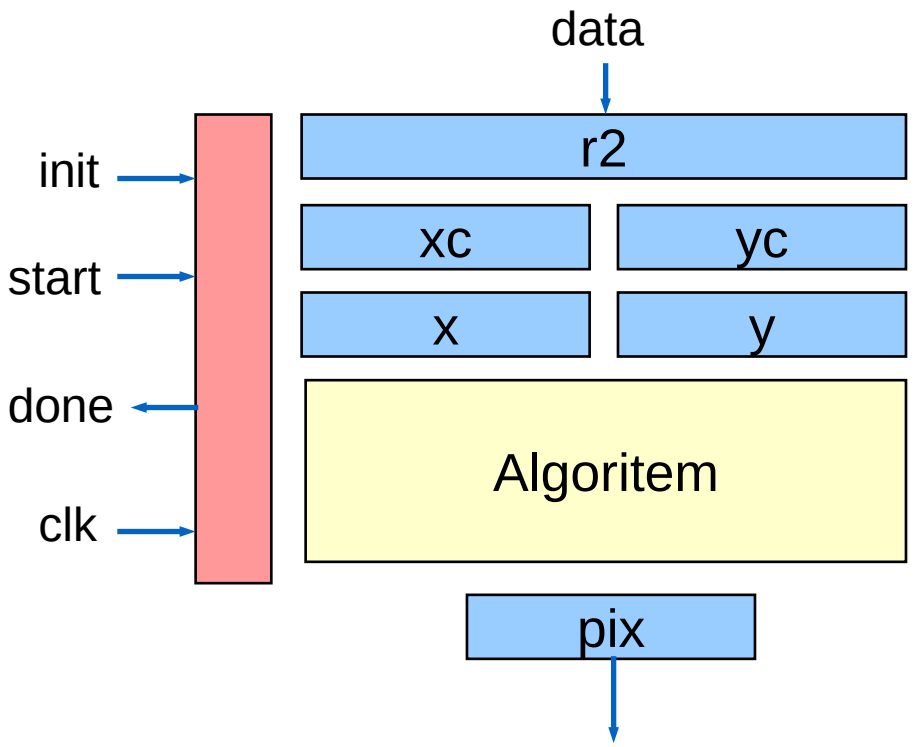

```
dx <= signed(x) - xc;
dy <= signed(y) - yc;

p1 <= dx * dx;
p2 <= dy * dy;

r <= p1 + p2;
```

```
if rising_edge(clk) then
  if r < r2 then
       pix <= '1';
  else
       pix <= '0';
  end if;
```

# Vmesnik vezja

Prek vmesnika nastavljamo parametre kroga in prenašamo podatke

Potrebujemo registre in krmilne signale

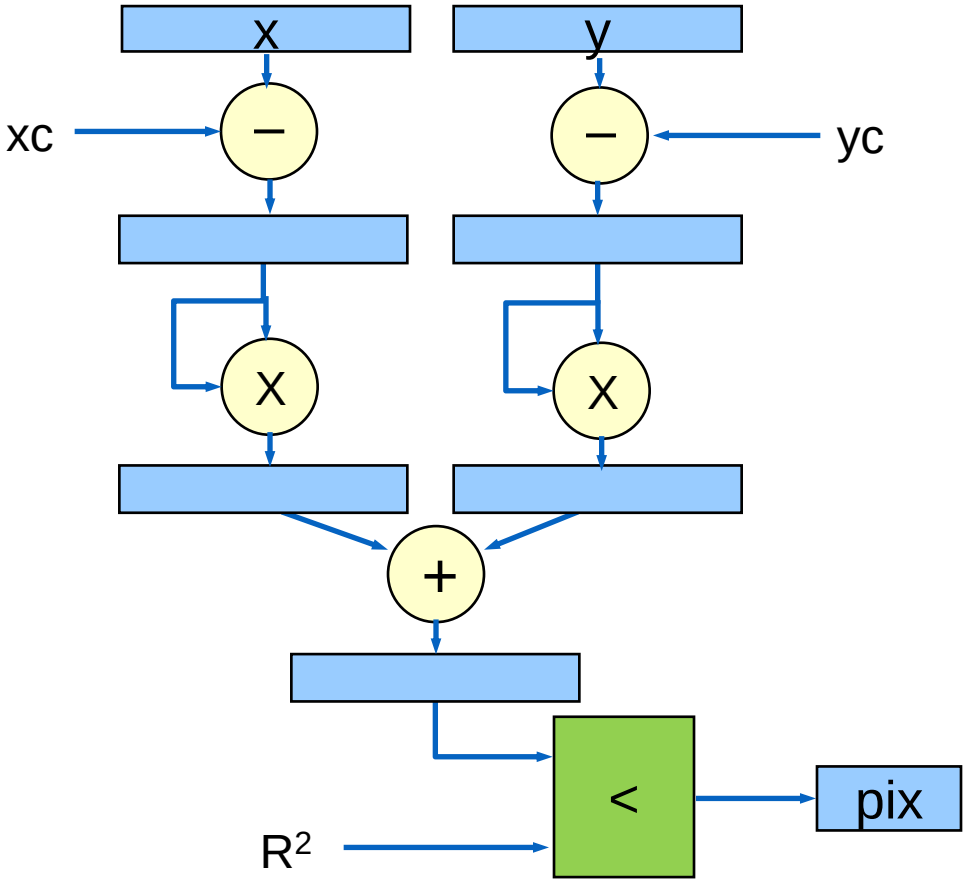



```
# FSMs                          : 1
# Multipliers                   : 2
 8x8-bit multiplier             : 2
# Adders/Subtractors            : 3
 16-bit adder                   : 1
 8-bit subtractor               : 2
# Registers                     : 51
 Flip-Flops                     : 51
# Comparators                   : 1
 16-bit comparator less         : 1
```

- Zasedenost vezja XC3S50A: 7% (52), 3% FF, 2/3 MULT
- zakasnitev: 1 cikel, prepustnost: 78 M op/s

# Optimizacija vezja – računski cevovod

Rezultat posamezne operacije shranimo v registru

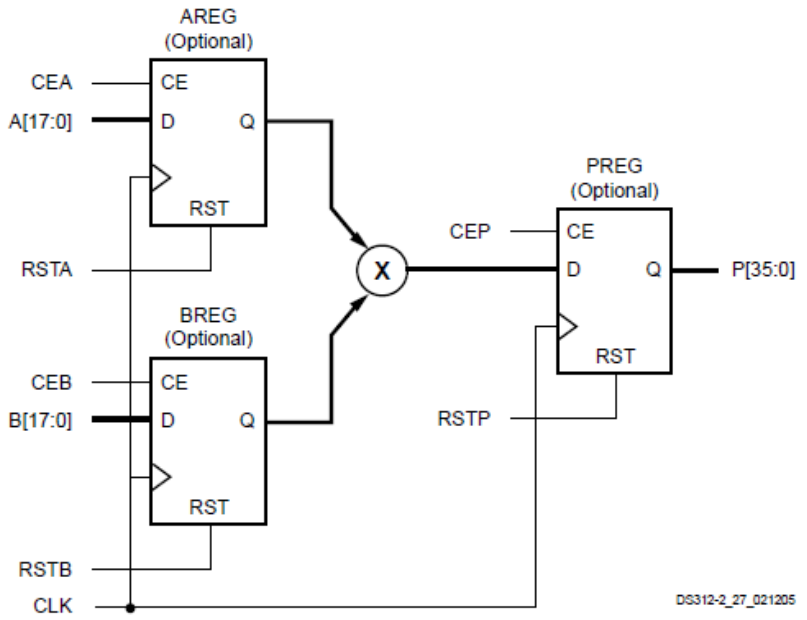



```
if rising_edge(clk) then
 dx <= signed(x) - xc;
 dy <= signed(y) - yc;

 p1 <= dx * dx;
 p2 <= dy * dy;

 r <= p1 + p2;
```

- cevovod: v vsakem ciklu obdela nov podatek
- zakasnitev (latenca) vezja je 4 cikle

# Rezultati sinteze in tehnološke preslikave

```
# FSMs                          : 1
# Multipliers                   : 2
 8x8-bit multiplier             : 2
# Adders/Subtractors            : 3
 16-bit adder                   : 1
 8-bit subtractor               : 2
# Registers                     : 67
 Flip-Flops                     : 67
# Comparators                   : 1
 16-bit comparator less         : 1
```

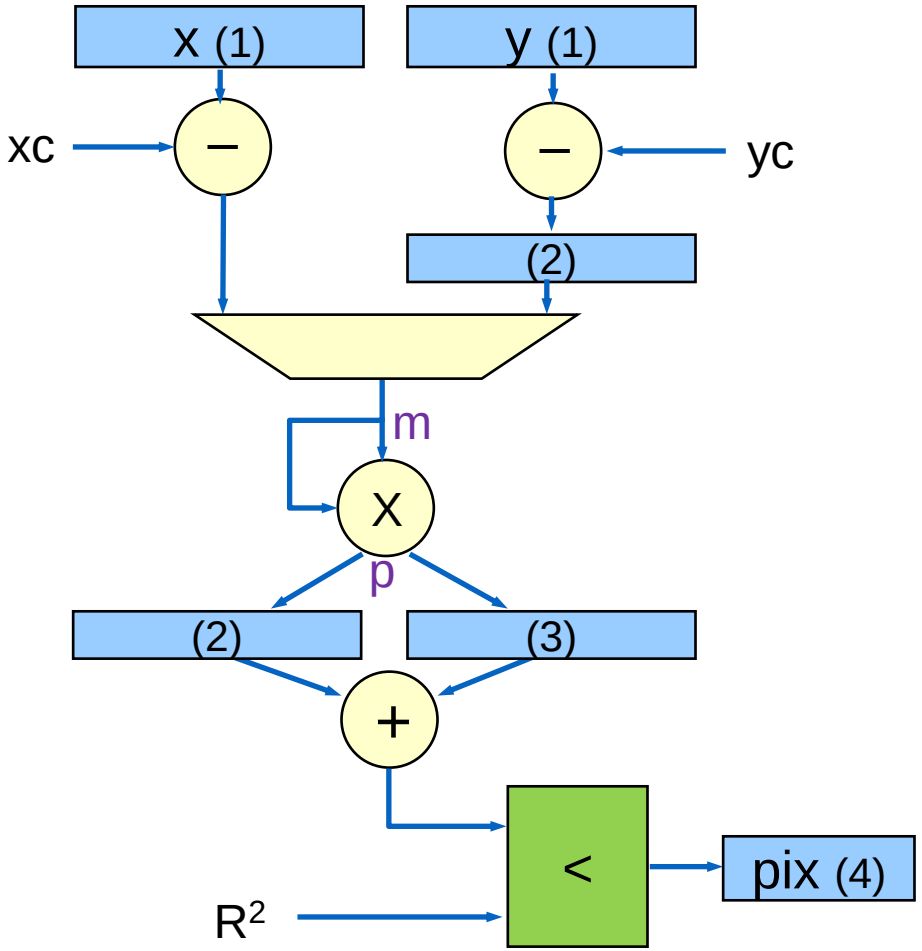


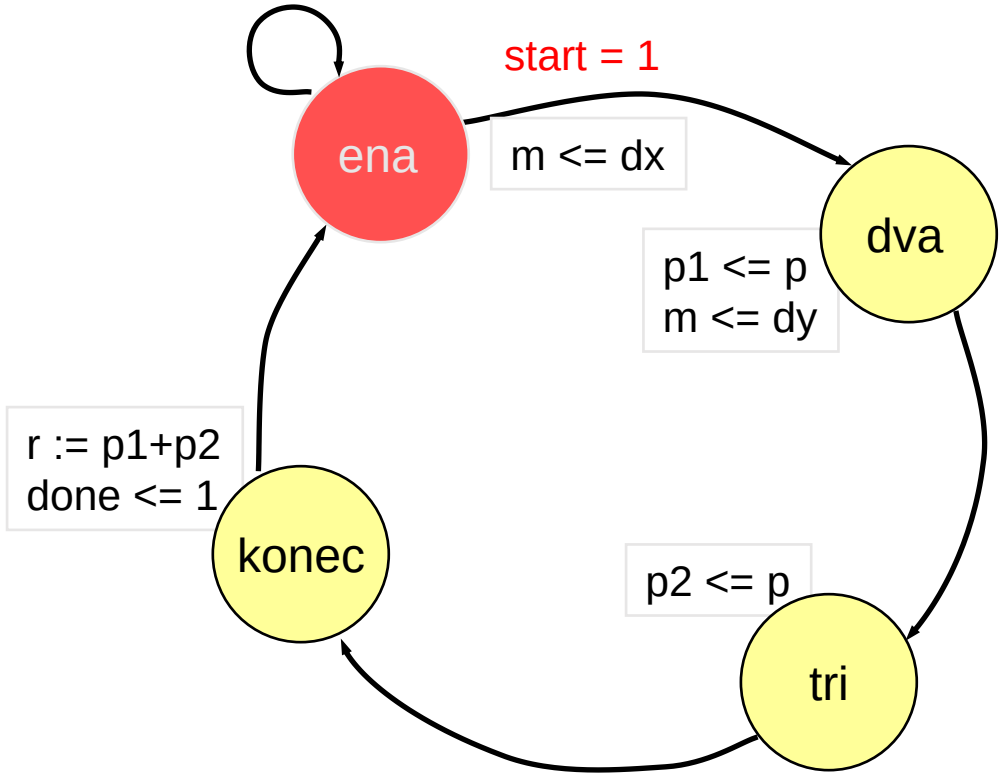

- Namesto 64 dodatnih FF (dx, dy, p1, p2, r) jih je le 16 !
  - produkt in registri so v posebnem bloku MULT znotraj FPGA
- Zasedenost vezja XC3S50A: 7% (53), 4% FF, 2/3 MULT
  - v celicah vezja FPGA je dovolj FF, zato se zasedenost ne poveča !
- zakasnitev: 4 cikle, prepustnost: 200 M op/s
  - rezultat brez uporabe blokov MULT: 20%, 8% FF, 102 MHz

# Optimizacija površine – računska sekvenca

2x uporabimo en MULT

Načrtujemo krmilni avtomat



# *Opis vezja v jeziku VHDL

- sekvenčni krmilni avtomat:

```
sekv: process (clk)
  variable dx, dy: signed(7 downto 0);
  variable r: signed(15 downto 0);
begin
 if rising_edge(clk) then
   if stanje=ena and start='1' then
     dx := signed(x) - xc;
     dy := signed(y) - yc;
     m <= dx;   -- vhodi za prvi produkt
     stanje <= dva;
   end if;

   if stanje=dva then
     p1 <= p;   -- shrani produkt in pripravi
     m <= dy;  -- vhode za drugi produkt
     stanje <= tri;
   end if;

   if stanje=tri then
     p2 <= p;  -- shrani drugi produkt
     stanje<=konec;
   end if;

   if stanje=konec then
     r := p1 + p2; -- sestej in primerjaj
     if r < r2 then   pix <= '1';
     else pix <= '0'; end if;
     done <= '1'; -- signal za zakljucek
     stanje <= ena;
   end if;
```

- in kombinacijski množilnik:

```
p <= m * m;
```

# Rezultati sinteze in tehnološke preslikave

- Zasedenost vezja XC3S50A: 9% (67), 5% FF, 1/3 MULT
- zakasnitev: 4 cikle, prepustnost: 177 MHz/4 = 44 M op/s

```
# FSMs                          : 2
# Multipliers                   : 1
 8x8-bit multiplier             : 1
# Adders/Subtractors            : 3
 16-bit adder                   : 1
 8-bit subtractor               : 2
# Registers                     : 73
 Flip-Flops                     : 73
# Comparators                   : 1
 16-bit comparator less         : 1
```

- Sinteza brez blokov MULT: 15%, 6% FF, 98 MHz
- Primerjava s cevovodom brez MULT: 20%, 8% FF, 102 MHz

# 4. vezje: šifriranje podatkov

Blokovni algoritem s 4 iteracijami šifriranja

## C++

```
int main(void)
{
 char a, b, a_nov, b_nov;
 char k[] = {0xff,0x0f,0x03,0x30};
 cin >> a;
 cin >> b;
 for (int i=0; i<=3; i++) {
      a_nov = b;
      b_nov = (b + k[i]) ^ a;
      a = a_nov;
      b = b_nov;
 }
 cout << a << b;
}
```

## VHDL

### Behavioral

```
for i in 0 to 3 loop
   a_nov := b;
   b_nov := (b + k(i)) xor a;
   a := a_nov;
   b := b_nov;
end loop;
```

# Kombinacijska izvedba

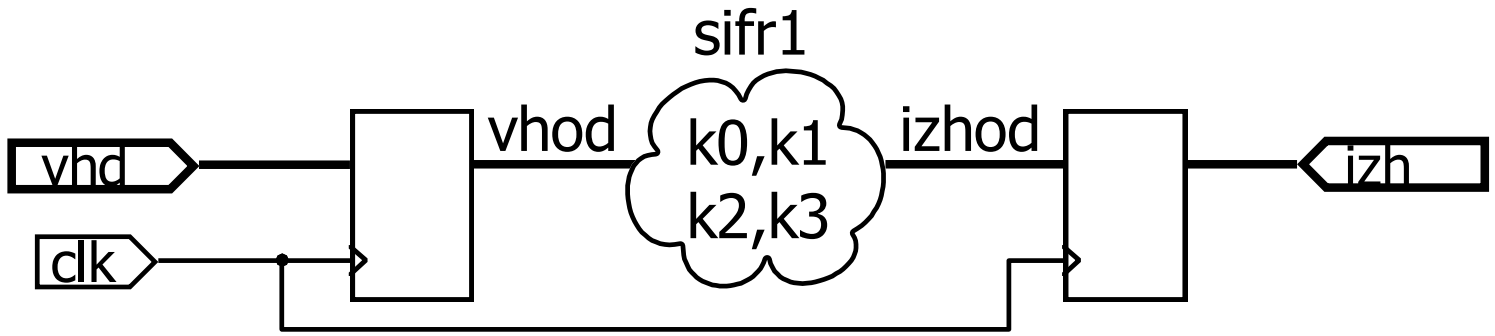
rezultat operacij vsakokrat shranimo v nov signal

```
sifr1: process(vhod, a0, b0, a1, b1, a2, b2, a3, b3, a4, b4 )
 begin
     a0 <= vhod(15 downto 8);
     b0 <= vhod(7 downto 0);
     a1 <= b0;
     b1 <= (b0 + k(0)) xor a0;
     a2 <= b1;
     b2 <= (b1 + k(1)) xor a1;
     a3 <= b2;
     b3 <= (b2 + k(2)) xor a2;
     a4 <= b3;
     b4 <= (b3 + k(3)) xor a3;
 end process;
```

**VHDL**

**RTL**

# Kombinacijsko šifriranje

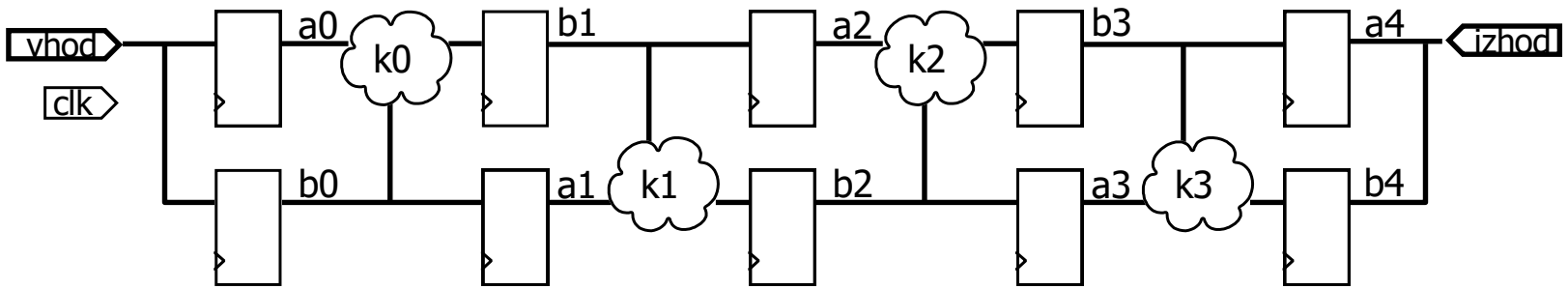



- zasedenost CPLD XC95288: 56 makrocelic, 32 flip-flopov
- zakasnitev: 1 cikel, prepustnost: 2*26,8 MHz = 53,6 MB/s

Kako povečati hitrost vezja?

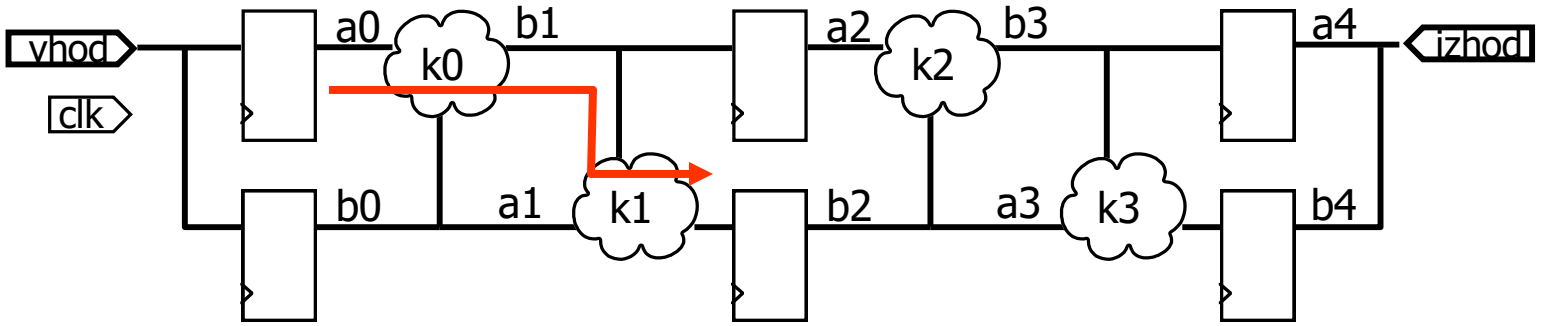
# Štiristopenjski cevovod

rezultat vsake iteracije shranimo v register




- zasedenost CPLD XC95288: 80 makrocelic, 80 flip-flopov
- zakasnitev: 4 cikle, prepustnost: 2*90,9 MHz = 181,8 MB/s

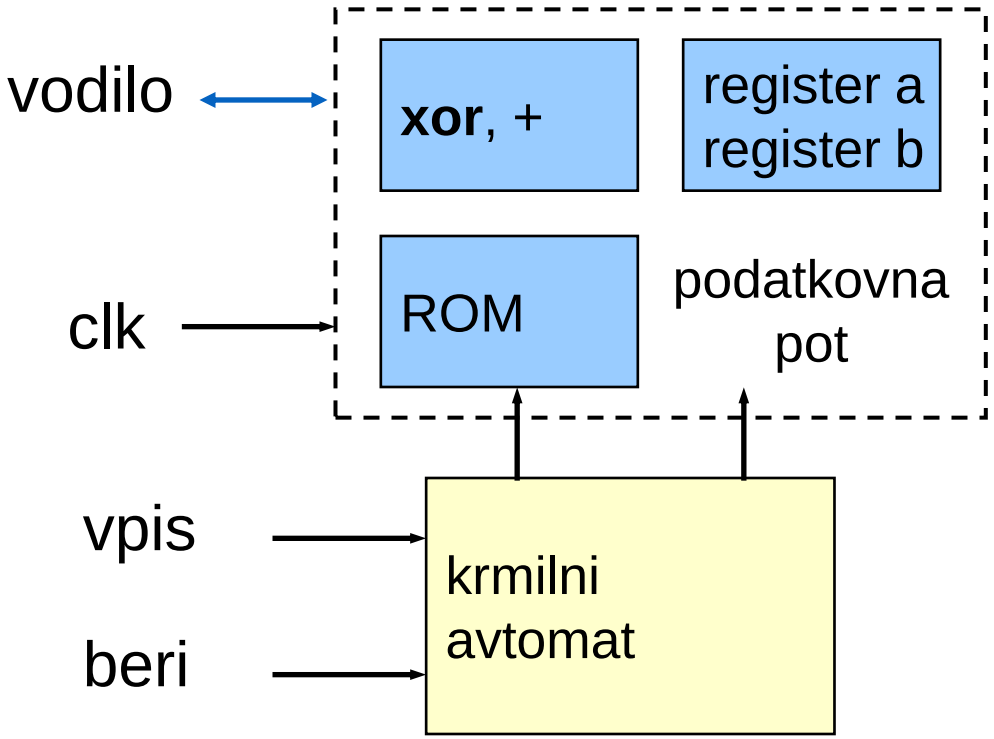
# Dvostopenjski cevovod

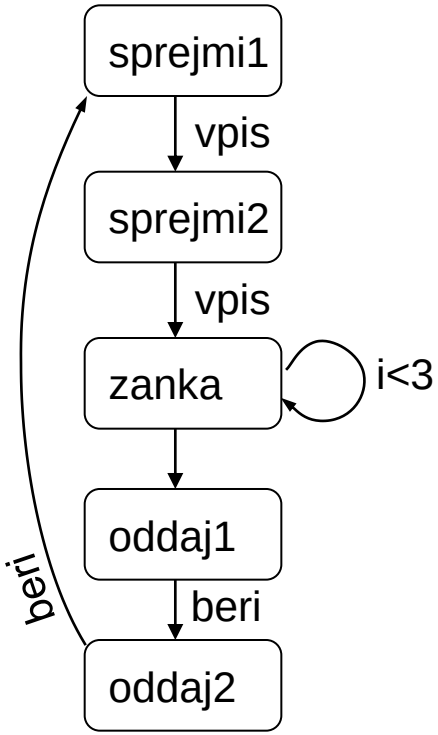



- zasedenost CPLD XC95288: 54 makrocelic, 48 flip-flopov
- zakasnitev: 2 cikla, prepustnost: 2*46,1 MHz = 92,2 MB/s

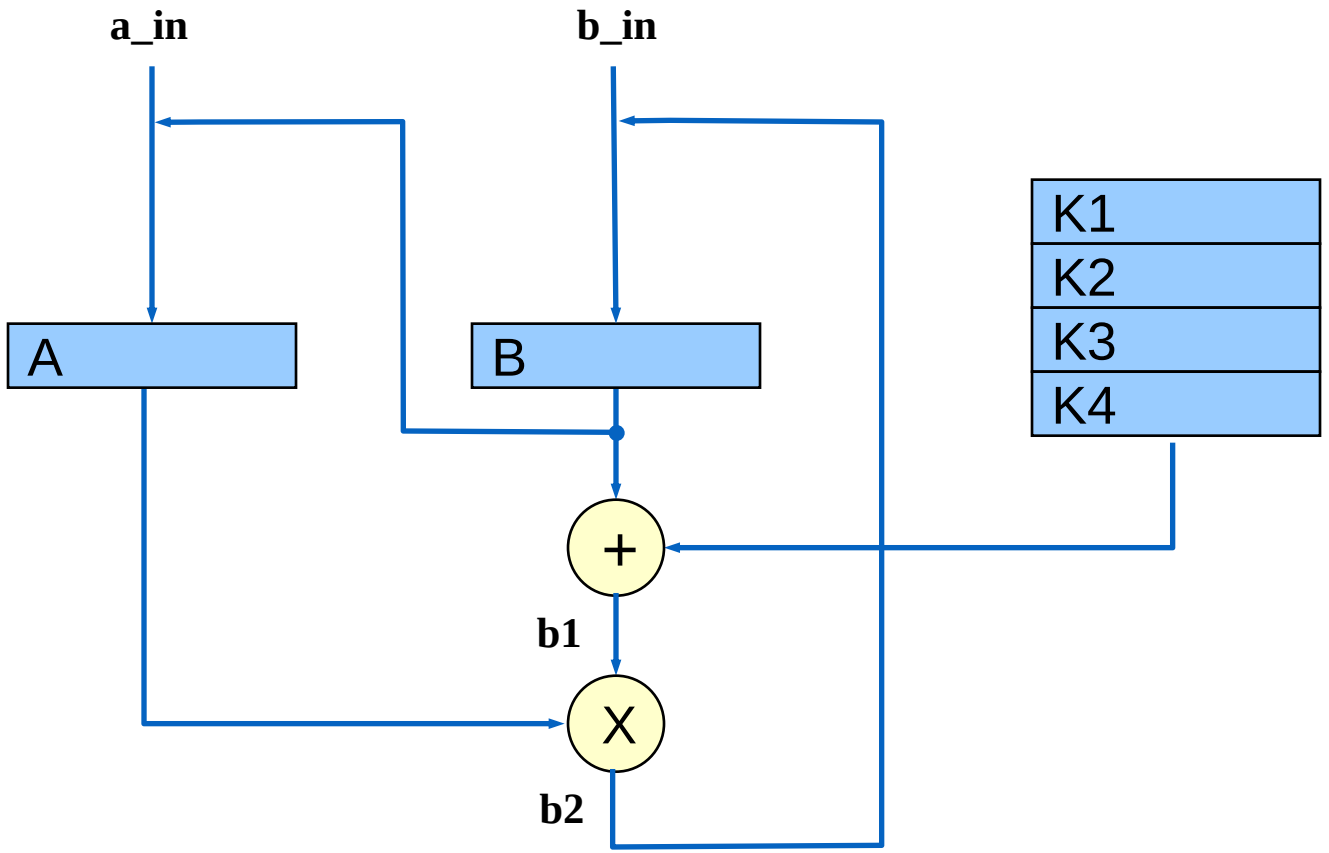
# Sekvenčni procesor

Vezje razdelimo na podatkovni in kontrolni del

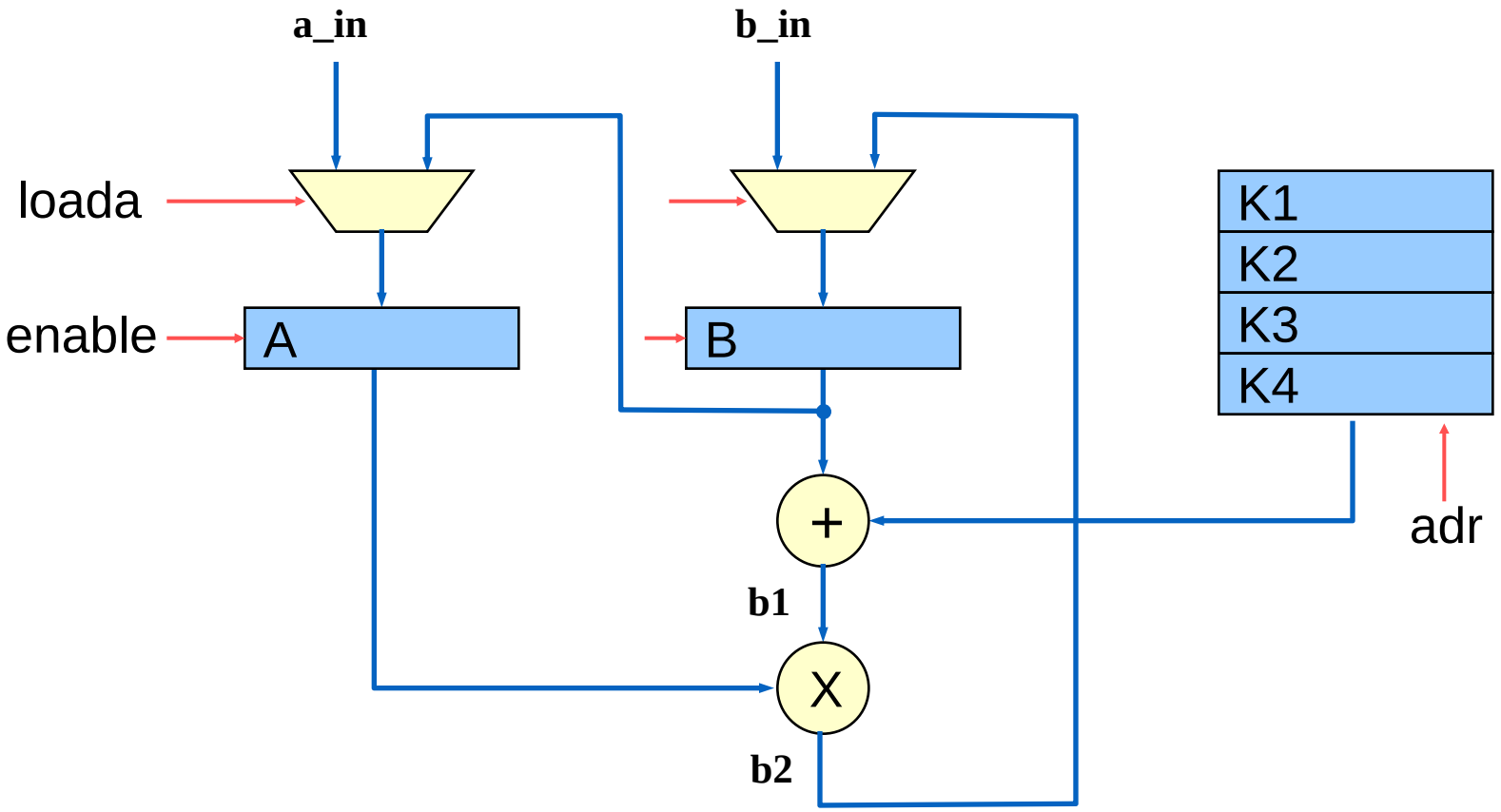


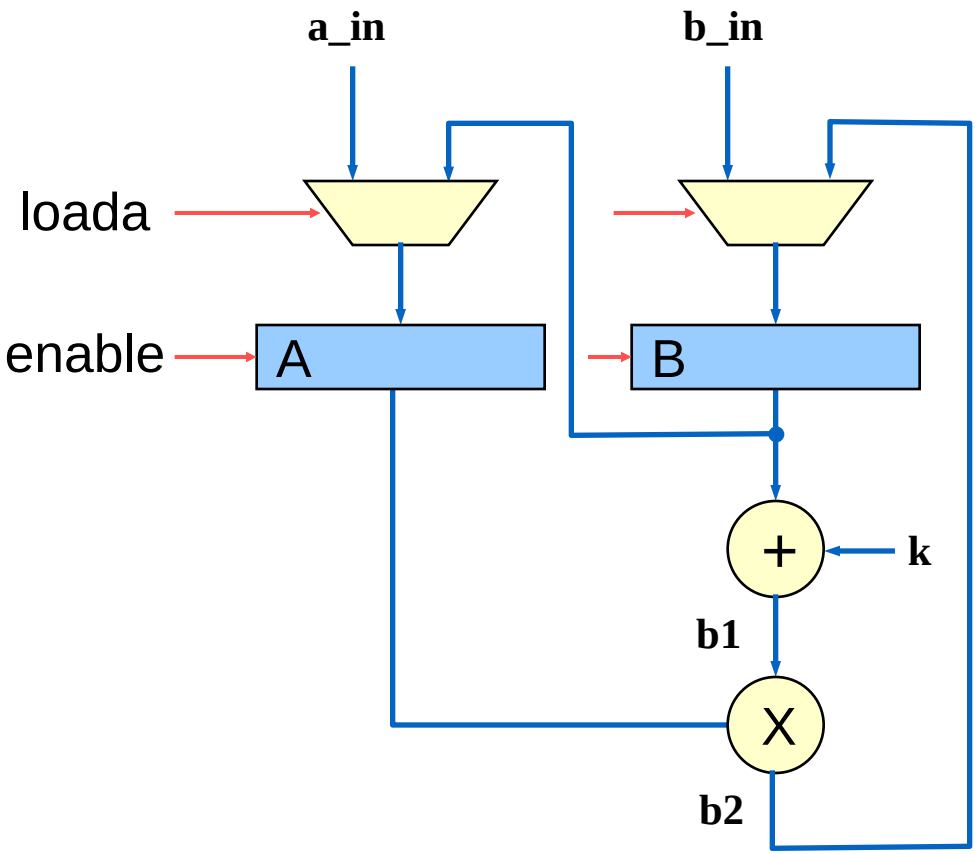
# Podatkovna pot



# Krmilni signali



# Podatkovna pot v jeziku VHDL

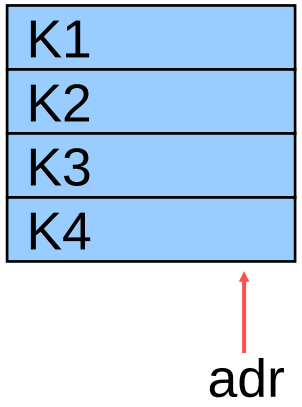



```
pod: process(clk)
begin
   if rising_edge(clk) then
      if enable = '1' then
         if loada = '1' then
            a <= a_in;
         elsif loadb = '1' then
            b <= b_in;
         else
            a <= b;
            b <= b2;
         end if;
      end if;
   end if;
end process;

b1 <= b + k;
b2 <= b1 xor a;
```

# ROM v jeziku VHDL

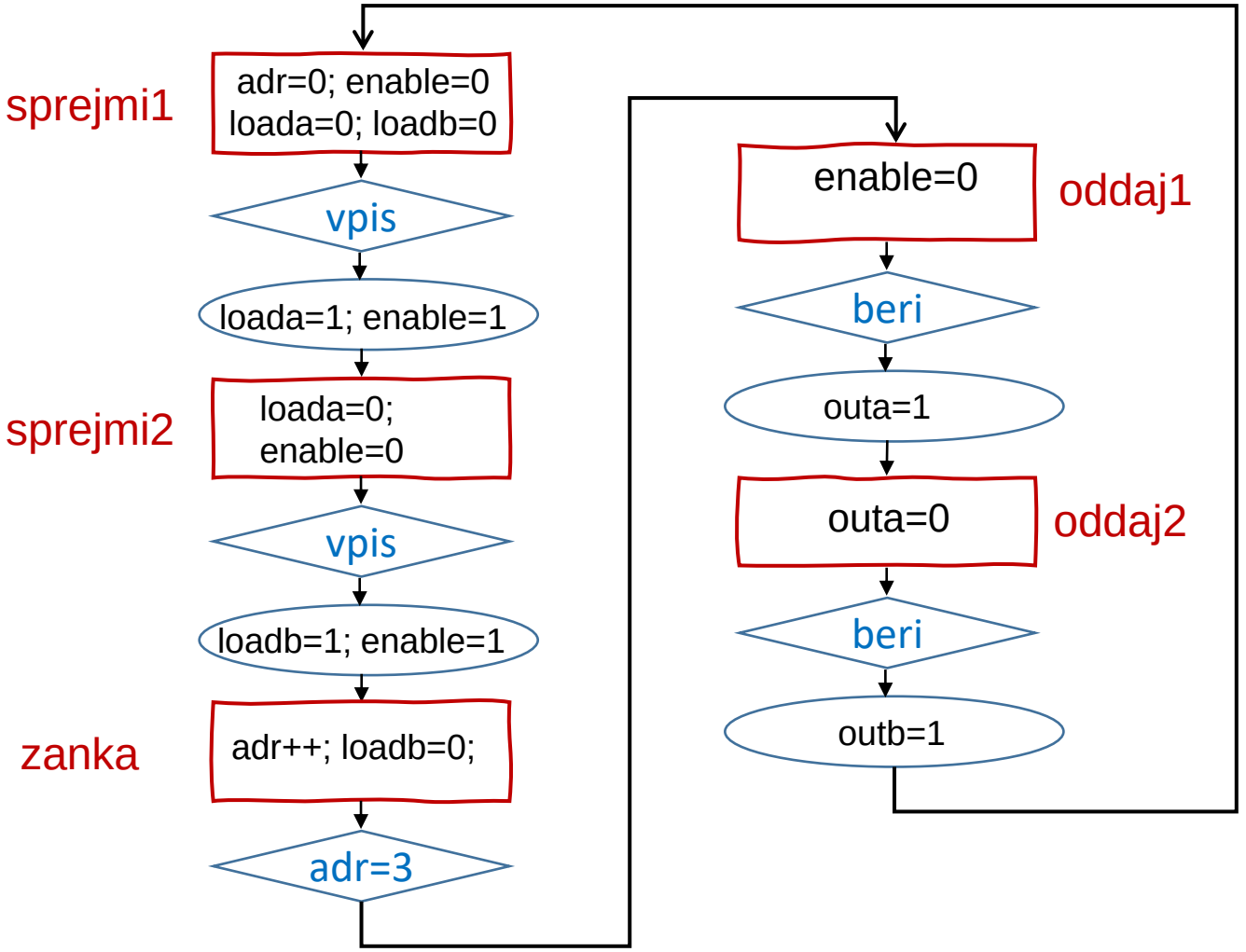
za opis uporabimo zbirko (array)

```
architecture RTL of code is
 ...
 type rom_type is array (0 to 3) of
      std_logic_vector (15 downto 0);

 constant rom : rom_type :=
      ("0000000000000001",
      "0000000000000011",
      "0000000000000111",
      "0000000000001111");

 signal adr: integer range 0 to 3;
 ...
begin
  k <= rom(adr);
```



# Krmilno vezje

algoritmični sekvenčni stroj



# Rezultat in primerjava

- zasedenost CPLD XC95288: 31 makrocelic, 22 flip-flopov
- zakasnitev: 8 ciklov, prepustnost: 84,7 MHz/8 = 10,6 MB/s

| Vezje | zasedenost | f [MHz] | MB/s |
|---|---|---|---|
| 1. vezje | 56 | 26,8 | 53,6 |
| 2. vezje | 80 | 90,9 | 181,8 |
| 3. vezje | 54 | 46,1 | 92,2 |
| 4. vezje | 31 | 84,7 | 10,6 |