Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

Digitalni Elektronski Sistemi

Osnove jezika VHDL

3. del: RTL opis in optimizacija vezja

Signali v sinhronih vezjih

- S signali poimenujemo povezave v vezju
- Signali pri simulaciji:
  - ob izračunu izrazov se določi bodoča vrednost signala (dogodek)
  - vrednost se priredi, šele ko se dogodki izvršijo
- Signali pri sintezi:
  - v sinhronih procesih (I. ali II. oblika) vsak signal, ki mu prirejamo vrednost pomeni register ali flip-flop
  - signali niso primerni za izračun vmesnih rezultatov s kombinacijsko logiko

Spremenljivke (variable)

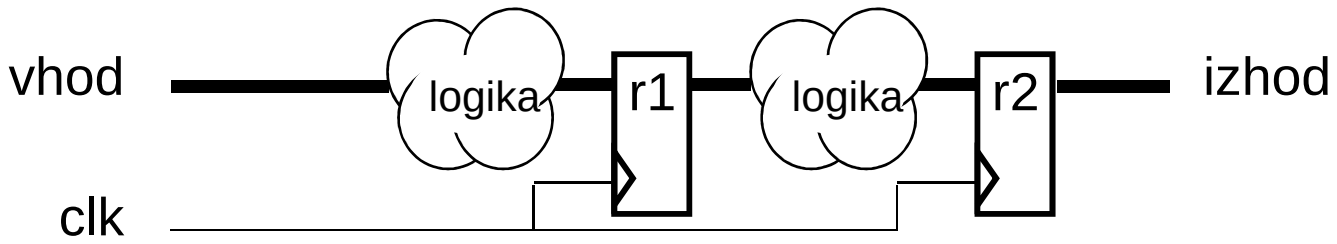
- Spremenljivke uporabljamo v procesnem okolju za izračun vmesnih rezultatov
  - ob izračunu spremenljivka takoj dobi novo vrednost
- Deklaracija spremenljivke:

```
p: process (clk)
  variable v: std_logic_vector(7 downto 0);
begin
```

- Prireditveni operator (:=)

```
v := vhod - “0011”;
```

Sinteza vezja s spremenljivkami




```
pipe: process (clk)
  variable v: std_logic_vector (3 downto 0);
begin
   if rising_edge(clk) then
      v := vhod - “0011”;
      if v>10 then
         r1 <= v;
      else
         r1 <= “1010”;
      end if;
      r2 <= r1 xor “1010”;
   end if;
end process;
```

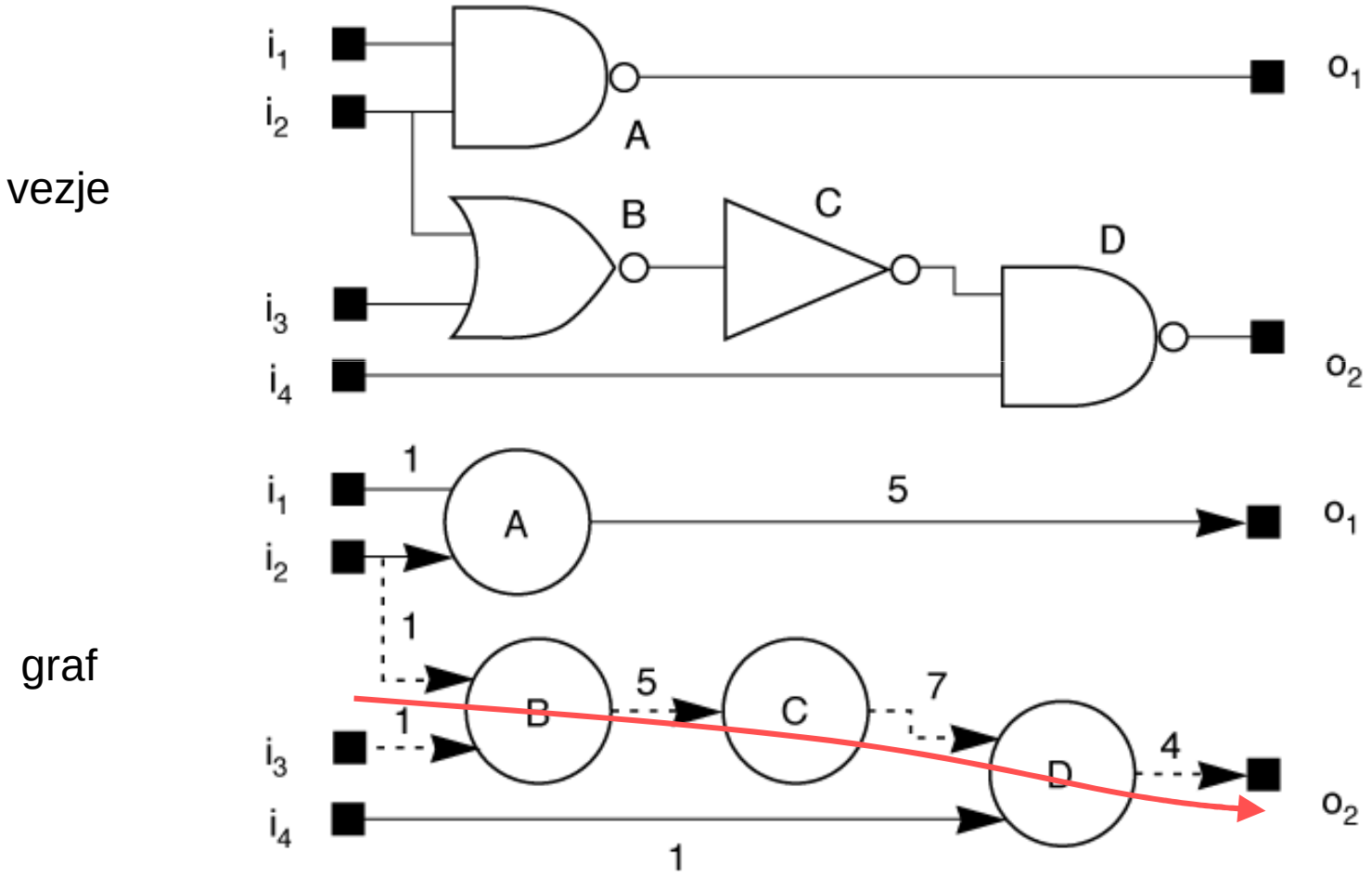
- spremenljivka ne predstavlja model povezave
- spremenljivka ne predstavlja registra

Zmogljivost vezja

- zmogljivost vezja omejujejo zakasnitve
- kombinacijske zakasnitve
  - od spremembe na vhodu do spremembe izhodov
  - poiščemo kritično pot in jo poskusimo skrajšati
    - lahko je več enakovrednih kritičnih poti
- ASIC: prevladujejo zakasnitve v logiki
  - zmanjšamo s topologijo vrat, obremenitvijo in načrtovanjem transistorjev
- FPGA / CPLD: velike zakasnitve v povezavah
  - izbira kratkih in namenskih povezav

Poti in zakasnitve

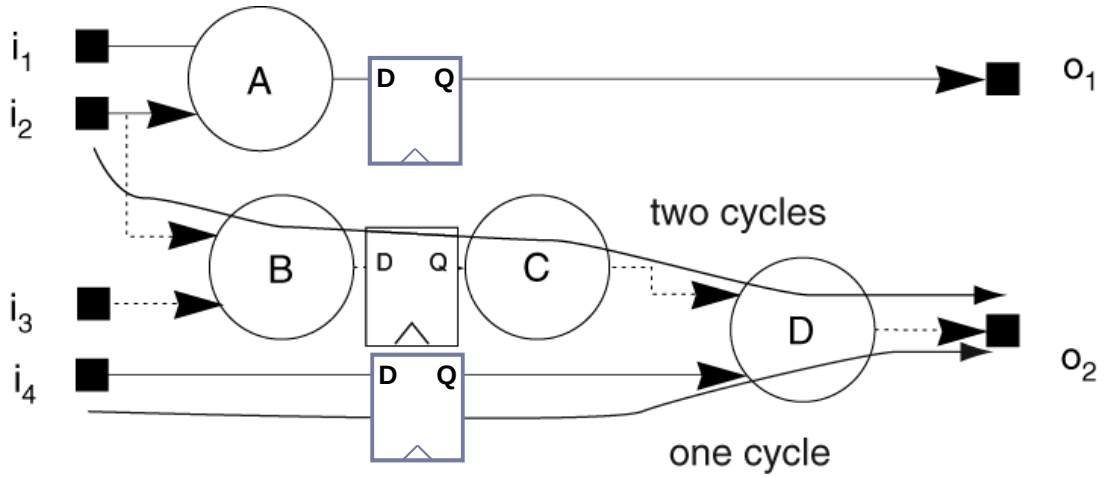
- kombinacijske zakasnitve merimo na poteh v vezju



- največja zakasnitev je na kritični poti

Optimizacija sekvenčnega vezja

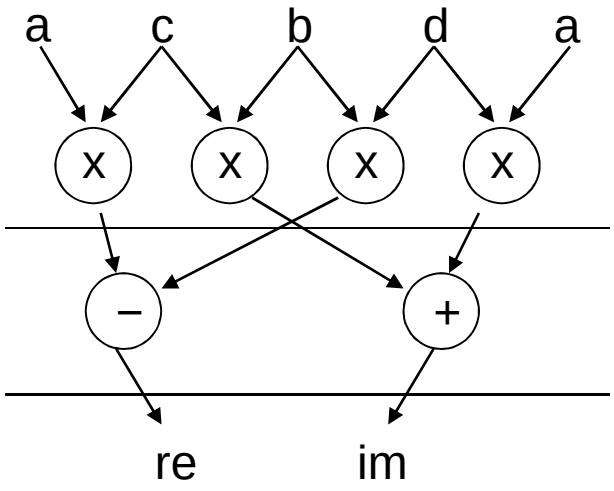
- Cevljenje (pipelining) z dodajanjem registrov
  - L : latenca (zakasnitev med vh. in izh.)
  - T : pretok (frekvenca obdelave podatkov)
- brez cevljenja
  - $L = D$
  - $T = 1 / D$
- dvostopenjski cevovod
  - $L = D + t_{FF}$
  - $T \leq 2 / D$



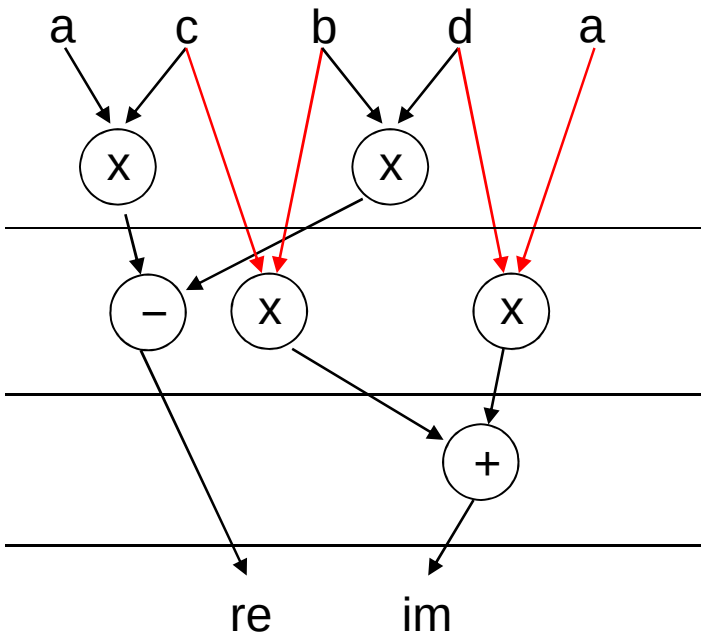
Optimizacija površine vezja

- Kompleksno množenje

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



- 4 mult, 2 addsub
- 2 računska cikla



- 2 mult, 1 addsub
- 3 računski cikli

Model vezja na nivoju RTL

- Algoritem določa obnašanje vezja
  - pri algoritmu ni določen časovni potek izvajanja
  - sinteza vezij iz algoritma je precej zahtevna
- Na nivoju RTL je določeno obnašanje vezja ob urinih ciklih
  - sinteza vezij iz RTL opisa je danes običajna
- Modeliramo prenos podatkov med pomnilnimi elementi in (kombinacijske) transformacije

Primer: 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;
```

1. vezje: razvijemo zanko

- kombinacijska izvedba zanke
  - rezultat operacij vsakokrat shranimo v nov signal

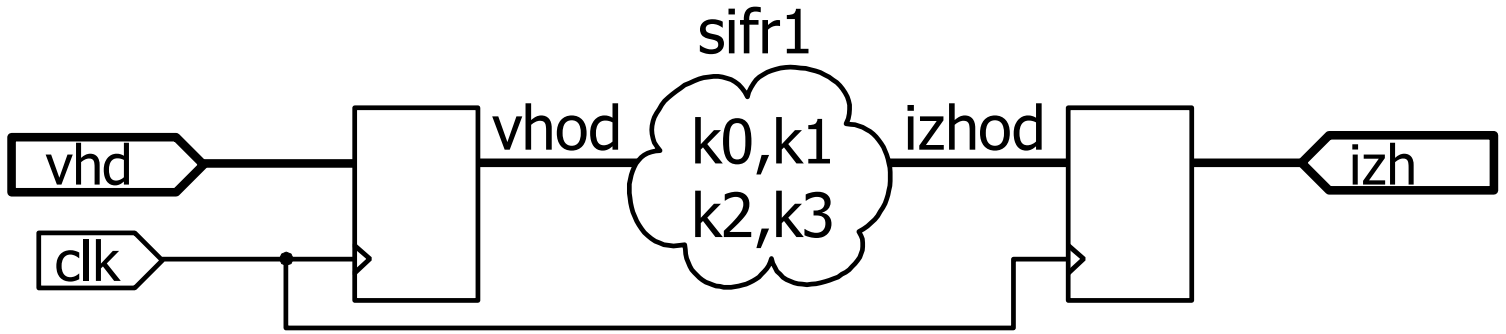
```
sifr1: process(vhod, a0, b0, a1, b1, a2, b2, a3, b3, a4, b4 )
 begin
     a0 <= vhod(15 downto 8);
     b0 <= vhod(7 downto 0);
     a1 <= b0;
     b1 <= (b0 + k(0)) xor a0;
     a2 <= b1;
     b2 <= (b1 + k(1)) xor a1;
     a3 <= b2;
     b3 <= (b2 + k(2)) xor a2;
     a4 <= b3;
     b4 <= (b3 +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

VHDL

RTL

Rezultat 1. vezja

- registri na vhodu in izhodu



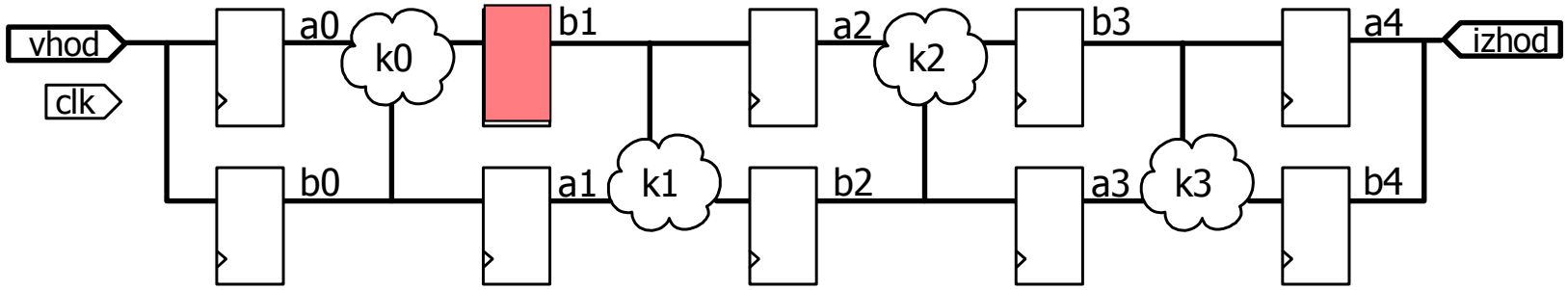
CPLD Xilinx XC95288XL-10

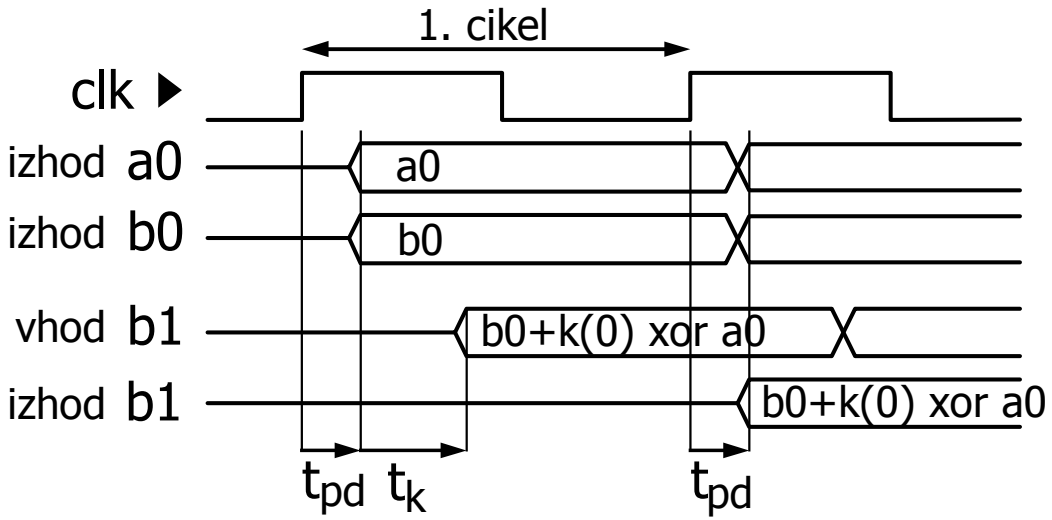
- Površina: 56 makrocelic, 32 flip-flopov
- Frekvenca ure: 26.8 MHz (53.6 MByte/s)

Kako povečati hitrost vezja?

2. vezje: polna cevovodna izvedba

- cevovodna izvedba zanke
  - rezultat vsake iteracije shranimo v register

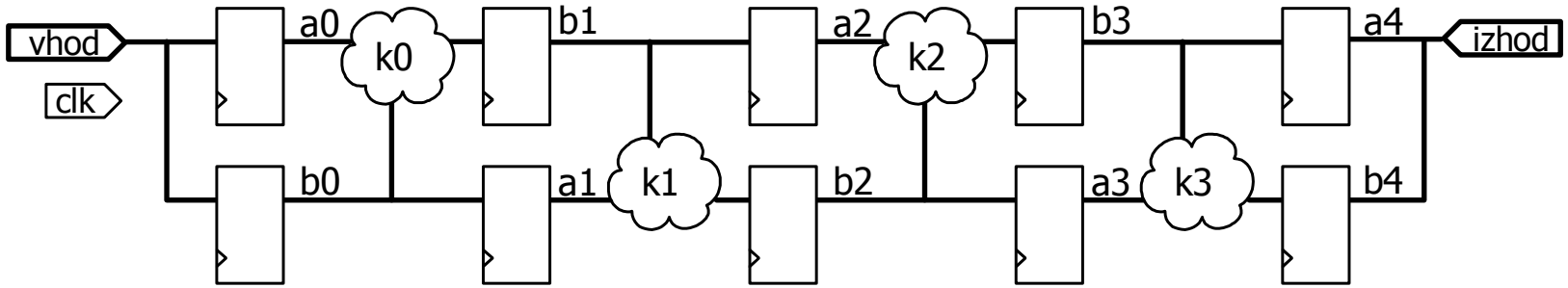




- opazujmo vhod in izhod registra b1
- frekvenca ure je omejena s $t_{pd}+t_k+t_s$
- vezje naj deluje le na eno fronto ure!

Rezultat 2. vezja

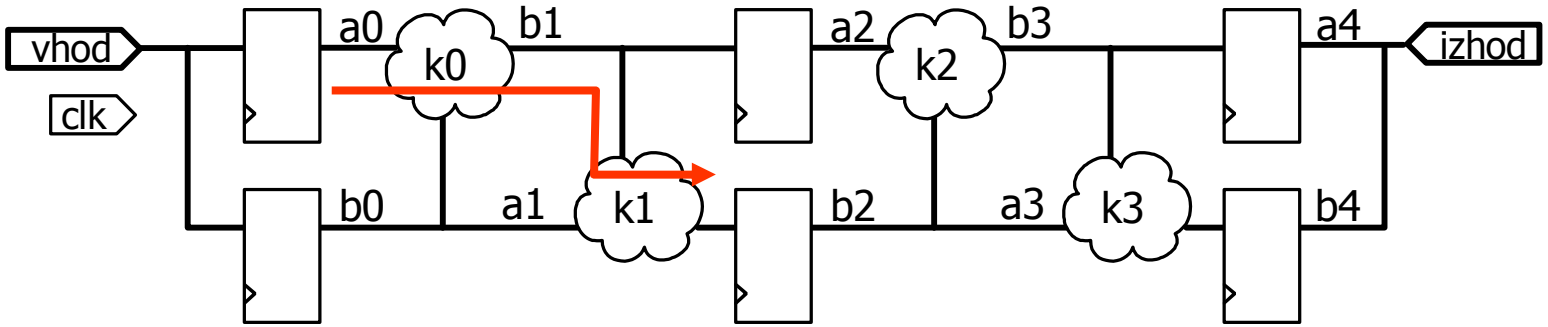
- 4-stopenjski cevovod



CPLD Xilinx XC95288XL-10

- Površina: 80 makrocelic, 80 flip-flopov
- Frekvenca ure: 90.9 MHz
- Zmogljivost: 181.8 MByte/s

3. vezje: dvostopenjski cevovod

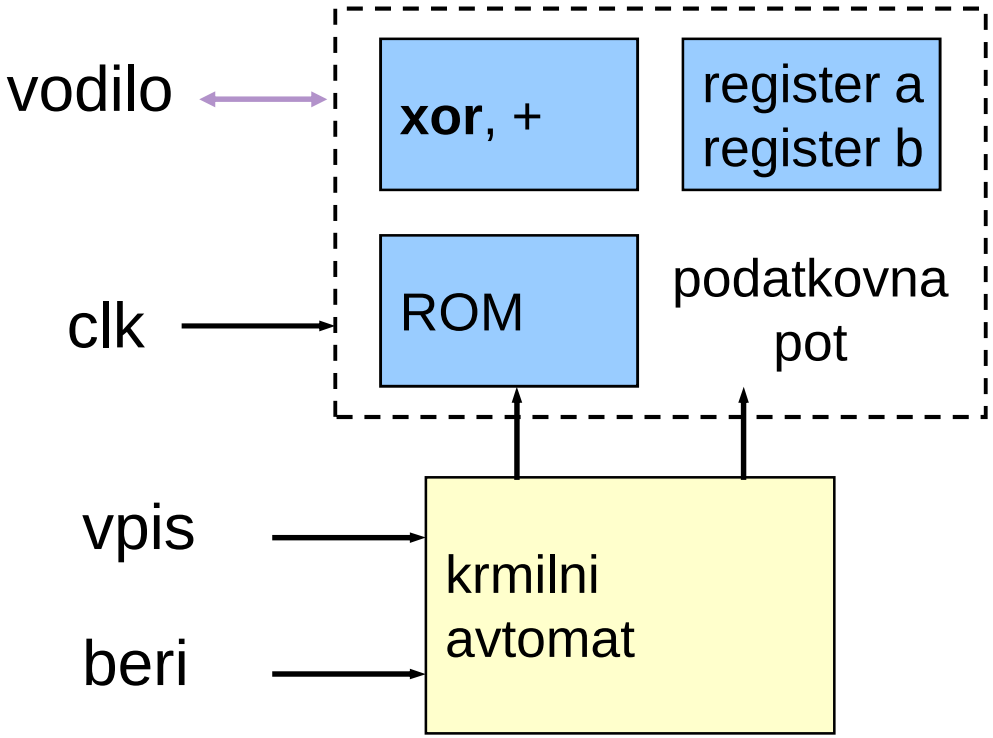


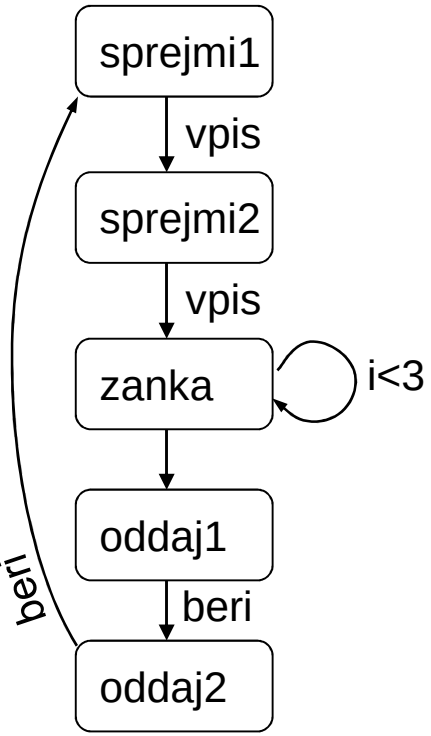
CPLD Xilinx XC95288XL-10

- Površina: 54 makrocelic, 48 flip-flopov
- Frekvenca ure: 46.1 MHz
- Zmogljivost: 92.2 MByte/s

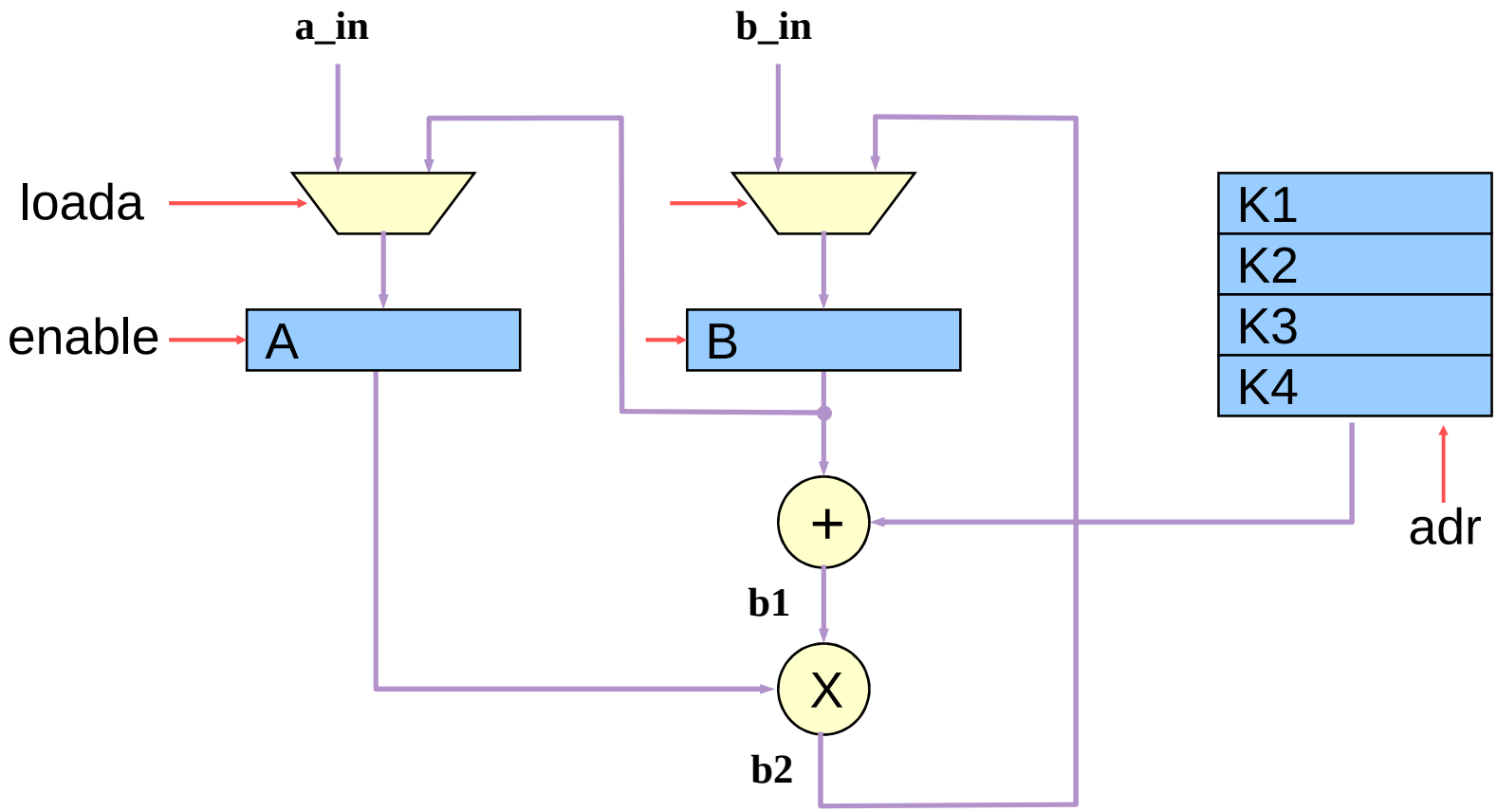
4. vezje: sekvenčni procesor

- Vezje razdelimo na podatkovni in kontrolni del

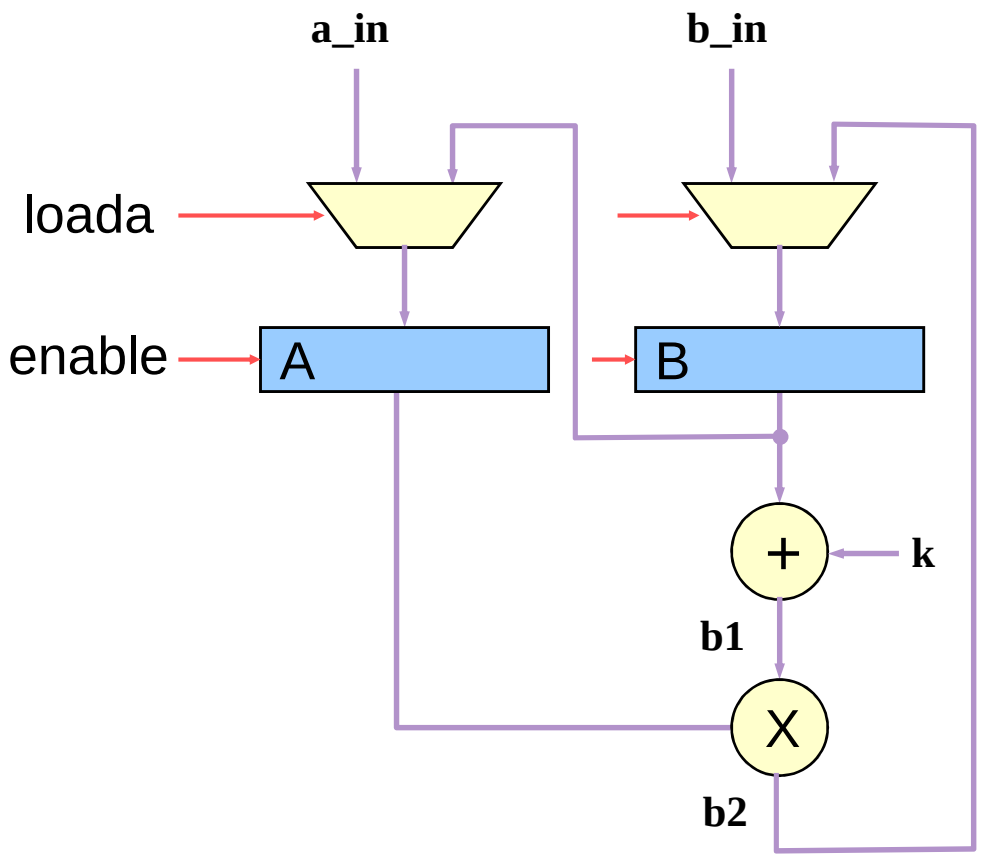




4. Krmilni signali in podatkovna pot



4. Podatkovna pot v jeziku VHDL

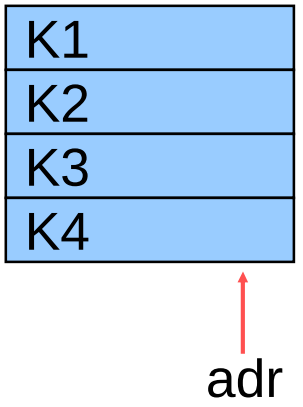



```
pod: process(clk)
begin
   if rising_edge(clk) then
      if enable = '1' then
         if loada = '1' then
            a <= a_in;
         elsif loadb = '1' then
            b <= b_in;
         else
            a <= b;
            b <= b2;
         end if;
      end if;
   end if;
end process;

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

4. ROM v jeziku VHDL

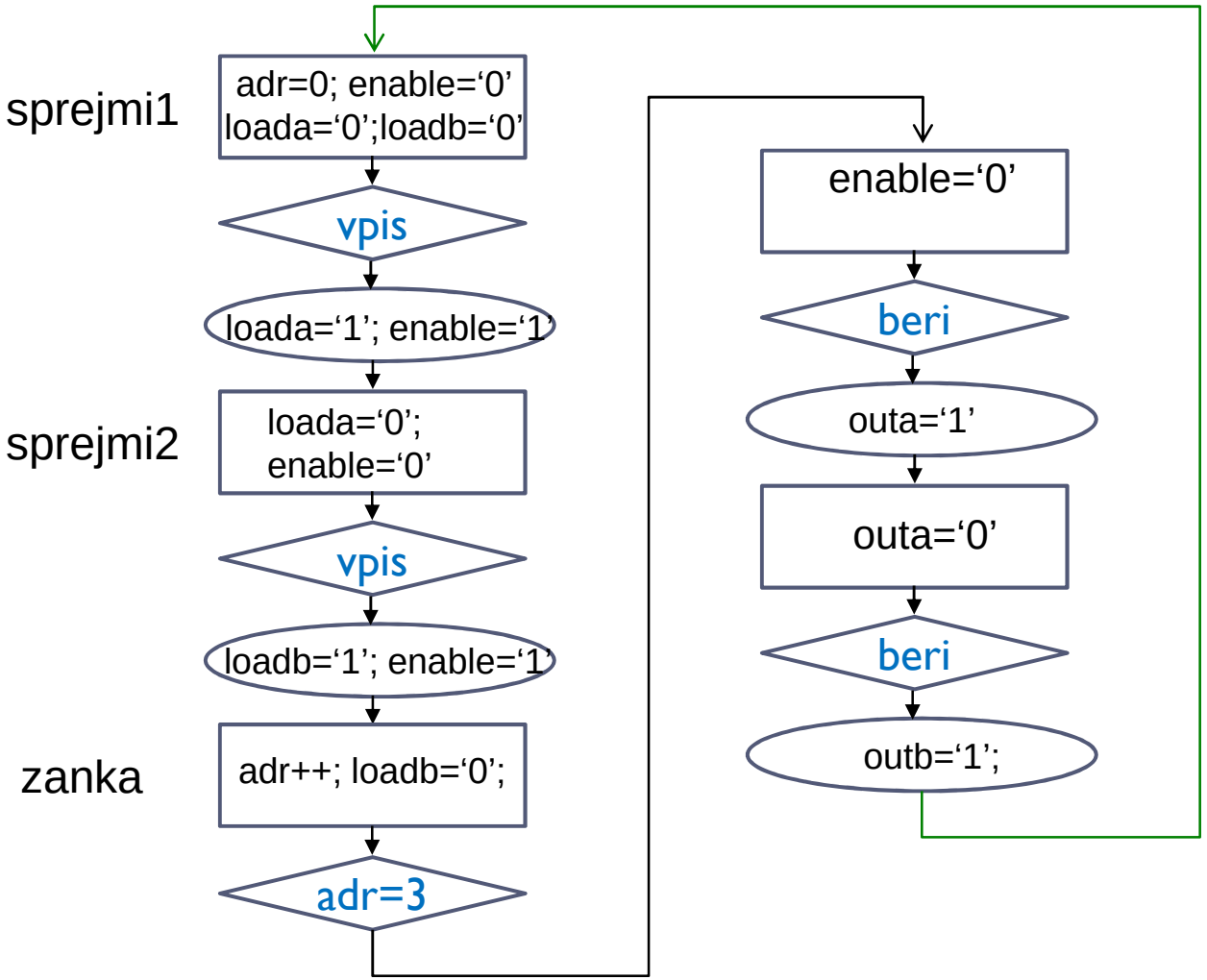
- za opis uporabimo zbirko (array)

```
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);
```



4. Krmilno vezje

- Algioritmični sekvenčni avtomat



Rezultat in primerjava

CPLD Xilinx XC95288XL-10

- Površina: 31 makrocelic, 22 flip-flopov
- Frekvenca ure: 84.7 MHz (10,6 Mbyte/s)

Vezje	Površina	f [MHz]	Mbyte/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