



Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*



Digitalni Elektronski Sistemi

Osnove jezika VHDL

Načrtovanje vezja na nivoju RTL

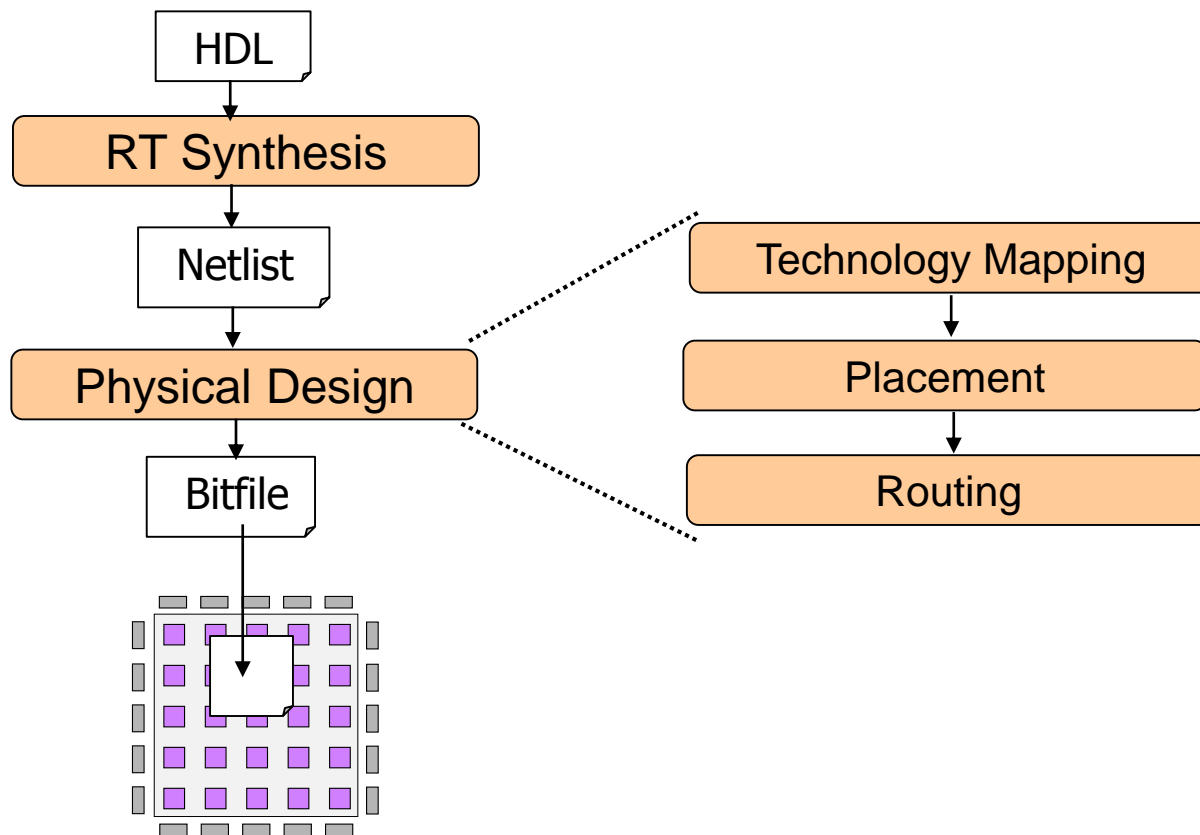
Model vezja na nivoju RTL

- ▶ Vezja za obdelavo podatkov izvajajo nek algoritem
 - ▶ algoritem opišemo kot zaporedje (računskih) korakov
- ▶ Algoritem določa obnašanje vezja
 - ▶ orodja za avtomatično sintezo vezja iz algoritma (C to RTL)
 - ▶ pri algoritmu ni določen časovni potek izvajanja
- ▶ Na nivoju RTL je določeno obnašanje vezja ob urnih ciklih
 - ▶ modeliramo prenos (transfer) in transformacijo podatkov med pomnilnimi elementi (registri): **RTL**
 - ▶ danes se večino digitalnih vezij načrtuje na nivoju RTL

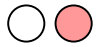
Prevajanje vezja na nivoju RTL

▶ RTL (Register Transfer Level)

- ▶ + omogoča natančno specifikacijo
- ▶ - časovno potratno, zahtevno kodiranje, več možnosti napak

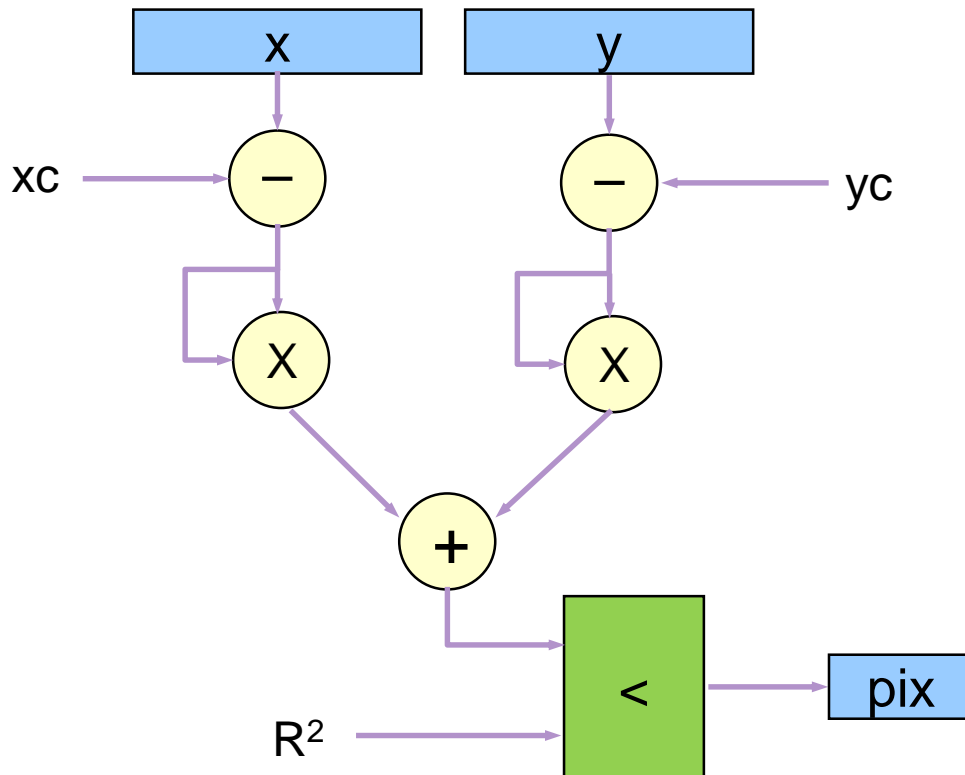


Primer algoritma – izračun točk v krogu



- ▶ Ali točka (x,y) leži znotraj krožnice (x_c, y_c, R^2) ?

$$(x-x_c)^2 + (y-y_c)^2 < R^2$$



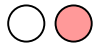
```
dx <= signed(x) - xc;
dy <= signed(y) - yc;
```

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

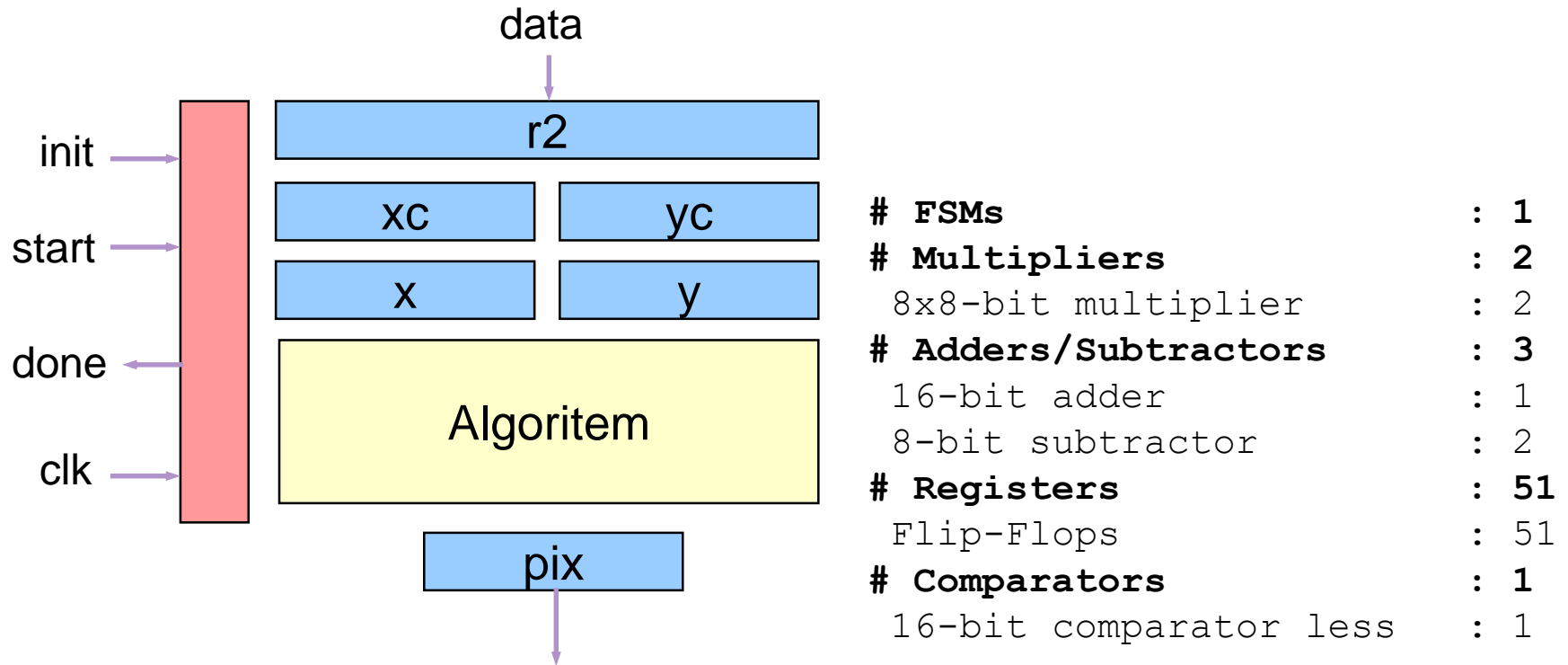
```
r <= p1 + p2;
```

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

Vmesnik vezja

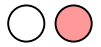


- ▶ Prek vmesnika nastavljamo parametre kroga in prenašamo podatke
- ▶ Potrebujemo registre in krmilne signale

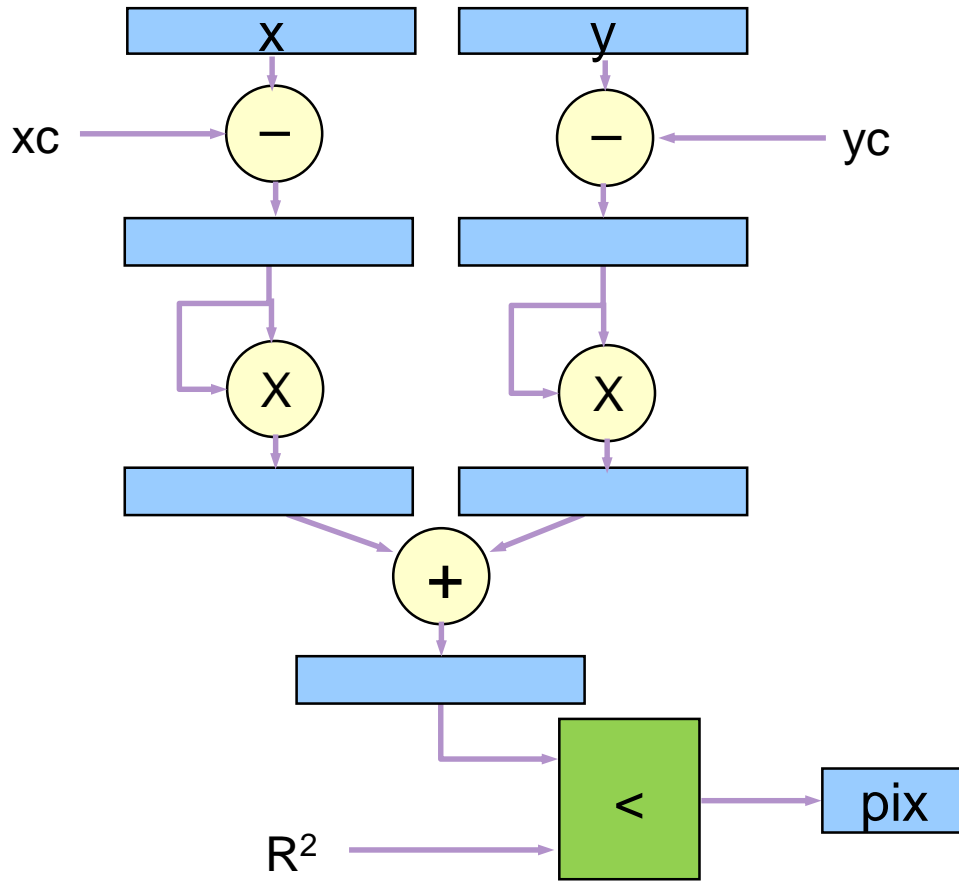


- ▶ Zasedenost vezja XC3S50A: 7% (52), 3% FF, 2/3 MULT
- ▶ Hitrost: max. frekvenca ure 78 MHz, 1 cikel računanja

Optimizacija vezja – računski cevovod



- ▶ Rezultat vsake operacije shranimo v registru
- ▶ manjše zakasnitve v posameznih korakih



```
if rising_edge(clk) then
```

```
  dx <= signed(x) - xc;
```

```
  dy <= signed(y) - yc;
```

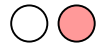
```
  p1 <= dx * dx;
```

```
  p2 <= dy * dy;
```

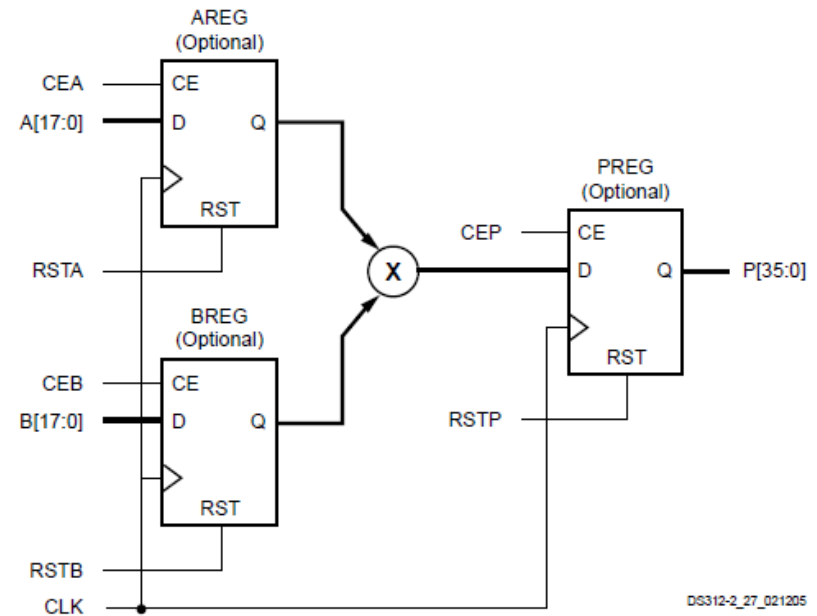
```
  r <= p1 + p2;
```

- ▶ cevovod omogoča, da pride vsak cikel nov podatek
- ▶ začetna zakasnitev (latenca) rezultata je 4 cikle

Rezultati sinteze in tehnološke preslikave

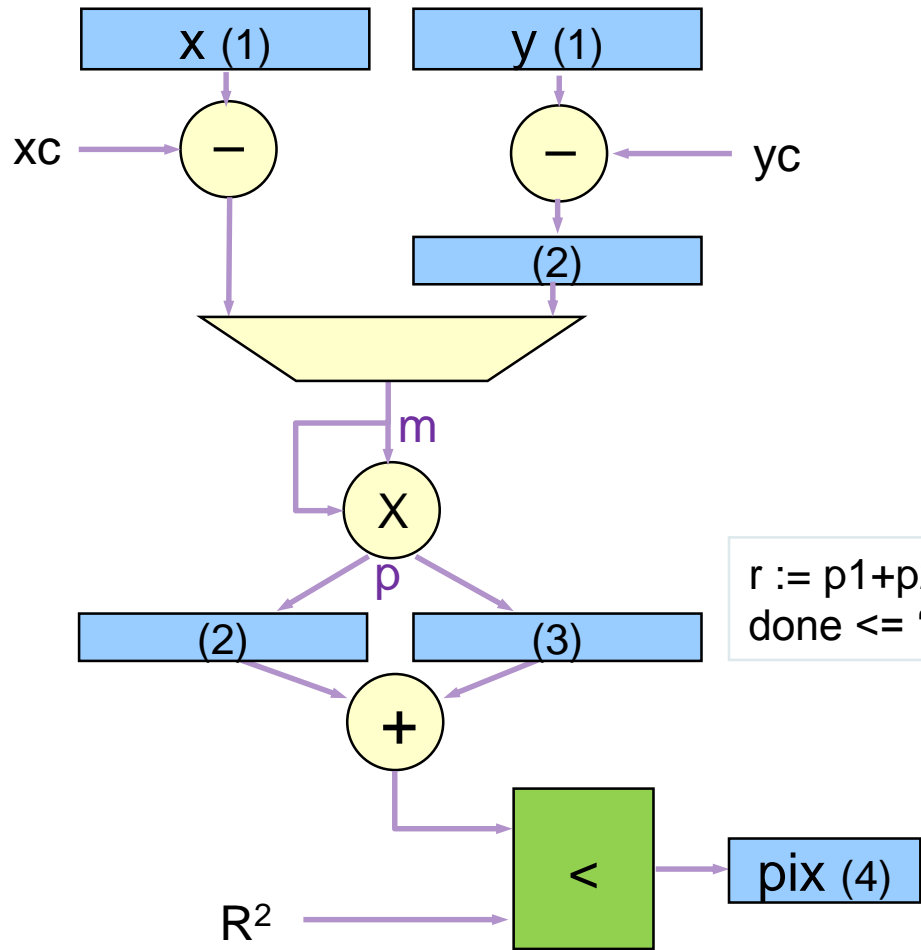


# FSMs	:	1
# Multipliers	:	2
8x8-bit multiplier	:	2
# Adders/Subtractors	:	3
16-bit adder	:	1
8-bit subtractor	:	2
# Registers	:	67
Flip-Flops	:	67
# Comparators	:	1
16-bit comparator less	:	1

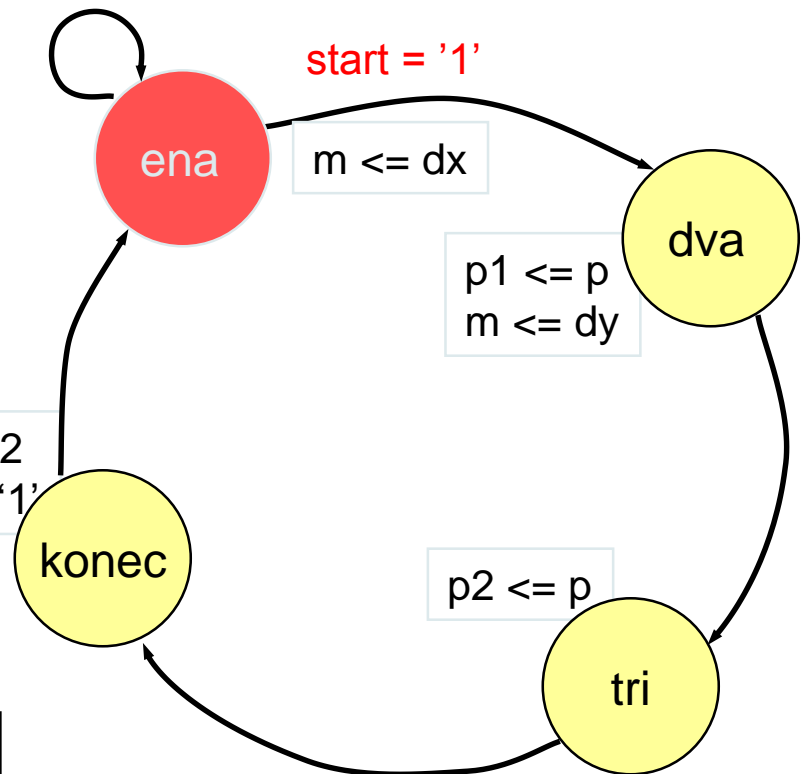


- ▶ Namesto 64 dodatnih FF (dx, dy, p1, p2, r) jih je le 16 !
 - ▶ produkt in registri so v posebnem bloku MULT znotraj FPGA
- ▶ Zasedenost vezja XC3S50A: 7% (53), 4% FF, 2/3 MULT
 - ▶ v celicah vezja FPGA je dovolj FF, zato se zasedenost ne poveča !
- ▶ Hitrost: max. frekvenca ure 200 MHz, 1(4) cikli računanja
- ▶ Rezultat brez uporabe blokov MULT: 20%, 8% FF, 102 MHz

Optimizacija površine – računska sekvenca ○●



- ▶ 2x uporabimo en MULT
- ▶ krmilni avtomat



► Kompakten opis sekvenčnega vezja

```
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;
```

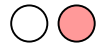
```
    if stanje=tri then
      p2 <= p;  -- shrani drugi produkt
      stanje <= konec;
    end if;

    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 zaključek
      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
- ▶ Hitrost: max. frekvenca ure 177 MHz, 4(4) cikli računanja

```
# 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
```

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

2. 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;
```

2.1 razvijemo zanko

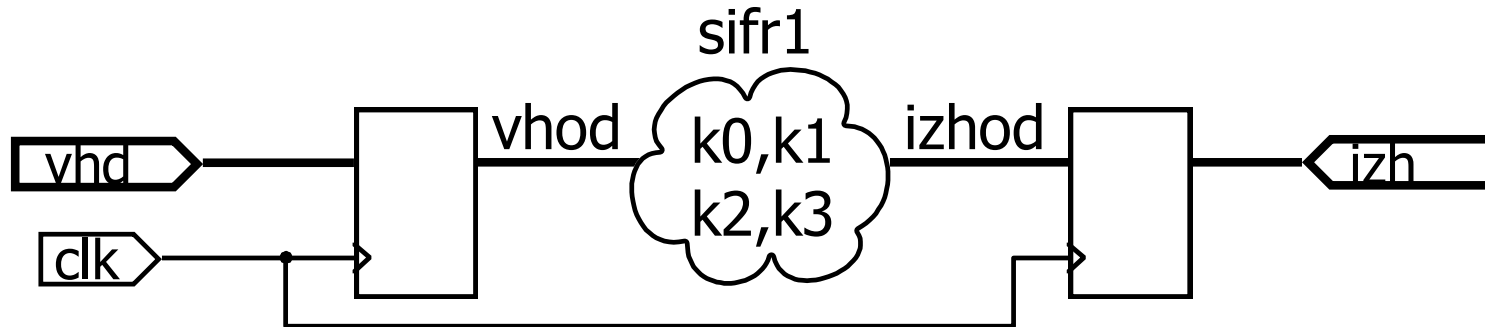


- ▶ kombinacijska izvedba zanke
 - ▶ rezultat operacij vsakokrat shranimo v nov signal

```
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

- registri na vhodu in izhodu

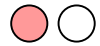


CPLD Xilinx XC95288XL-10

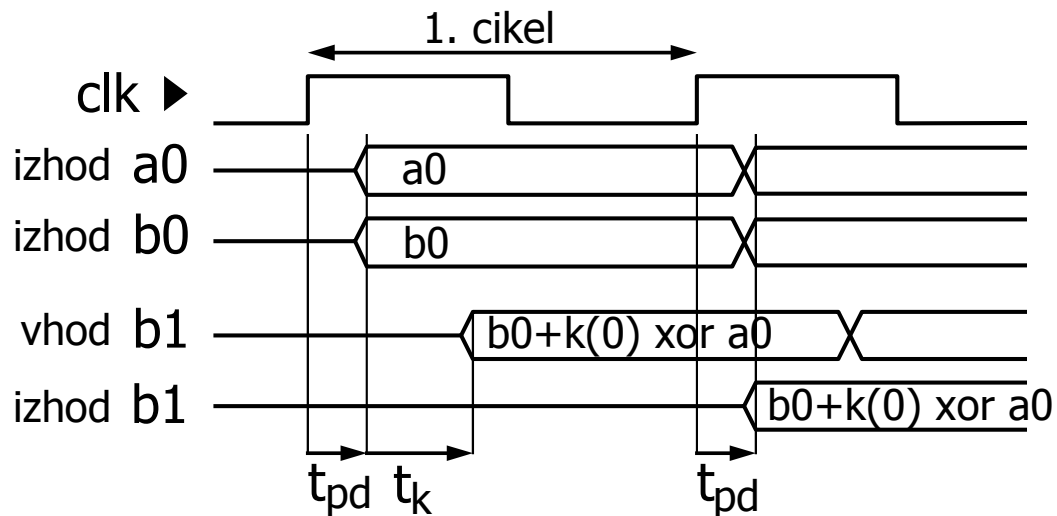
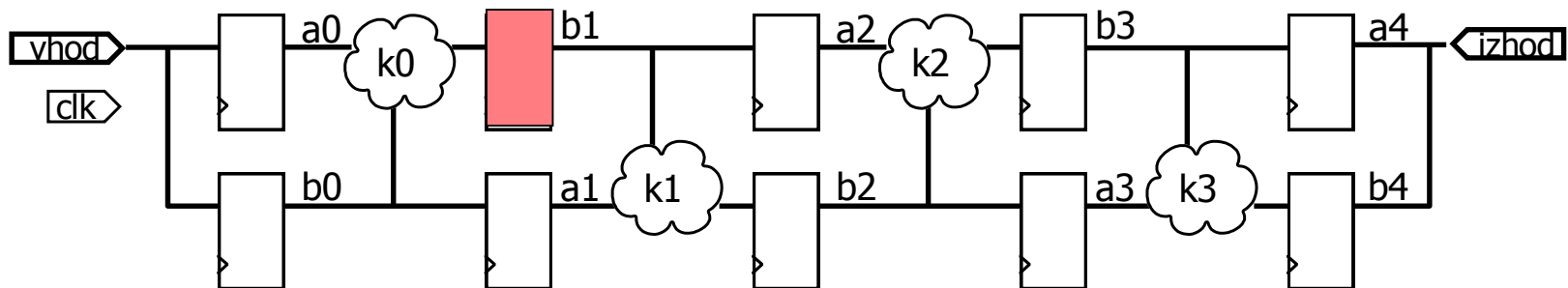
- ▶ Površina: 56 makrocelic, 32 flip-flopov
- ▶ Frekvenca ure: 26.8 MHz (53.6 MByte/s)

Kako povečati hitrost vezja?

2.2 polna cevovodna izvedba

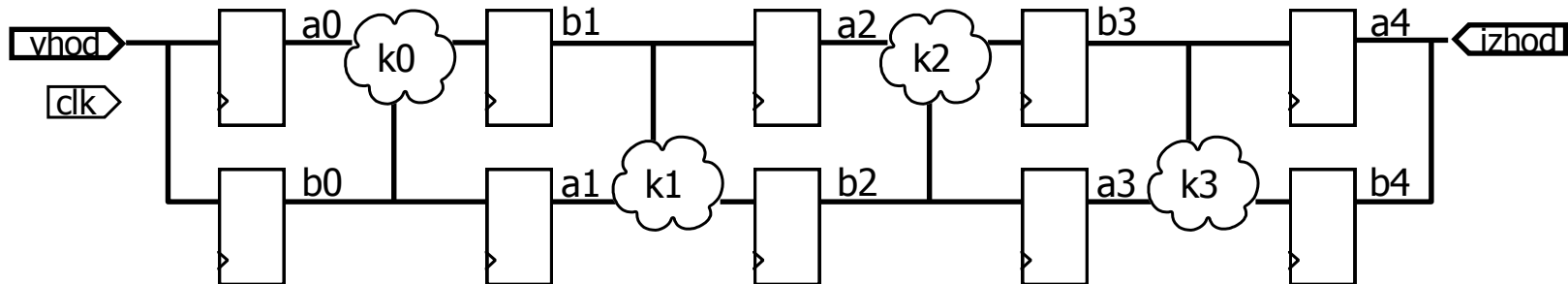


- cevovodna izvedba zanke
 - rezultat vsake iteracije shranimo v register



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

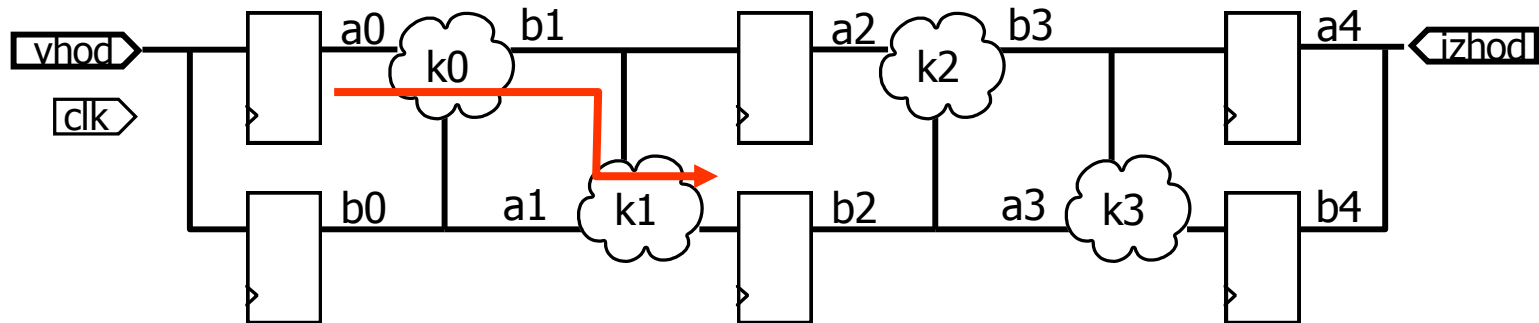
- 4-stopenjski cevovod



CPLD Xilinx XC95288XL-10

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

2.3: dvostopenjski cevovod



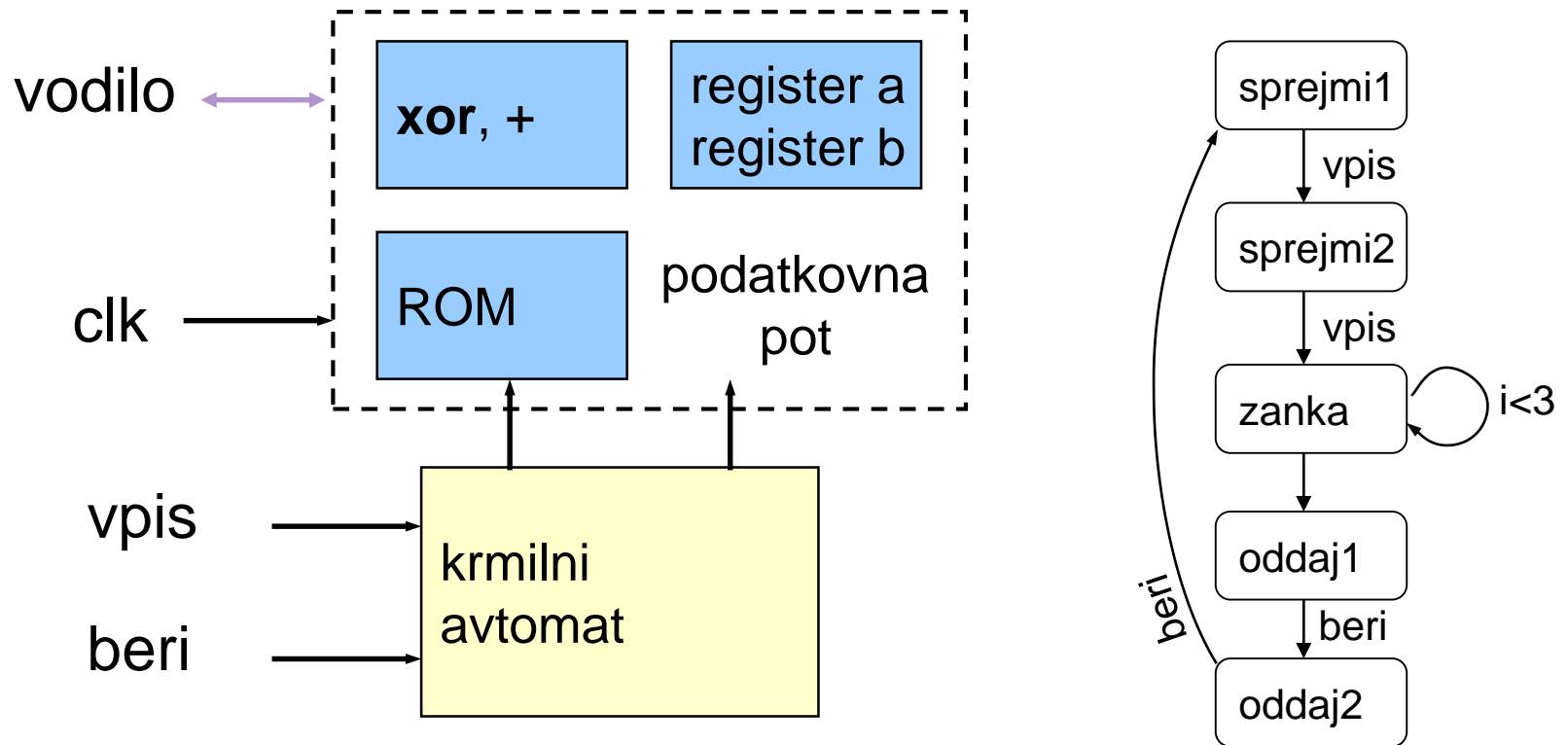
CPLD Xilinx XC95288XL-10

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

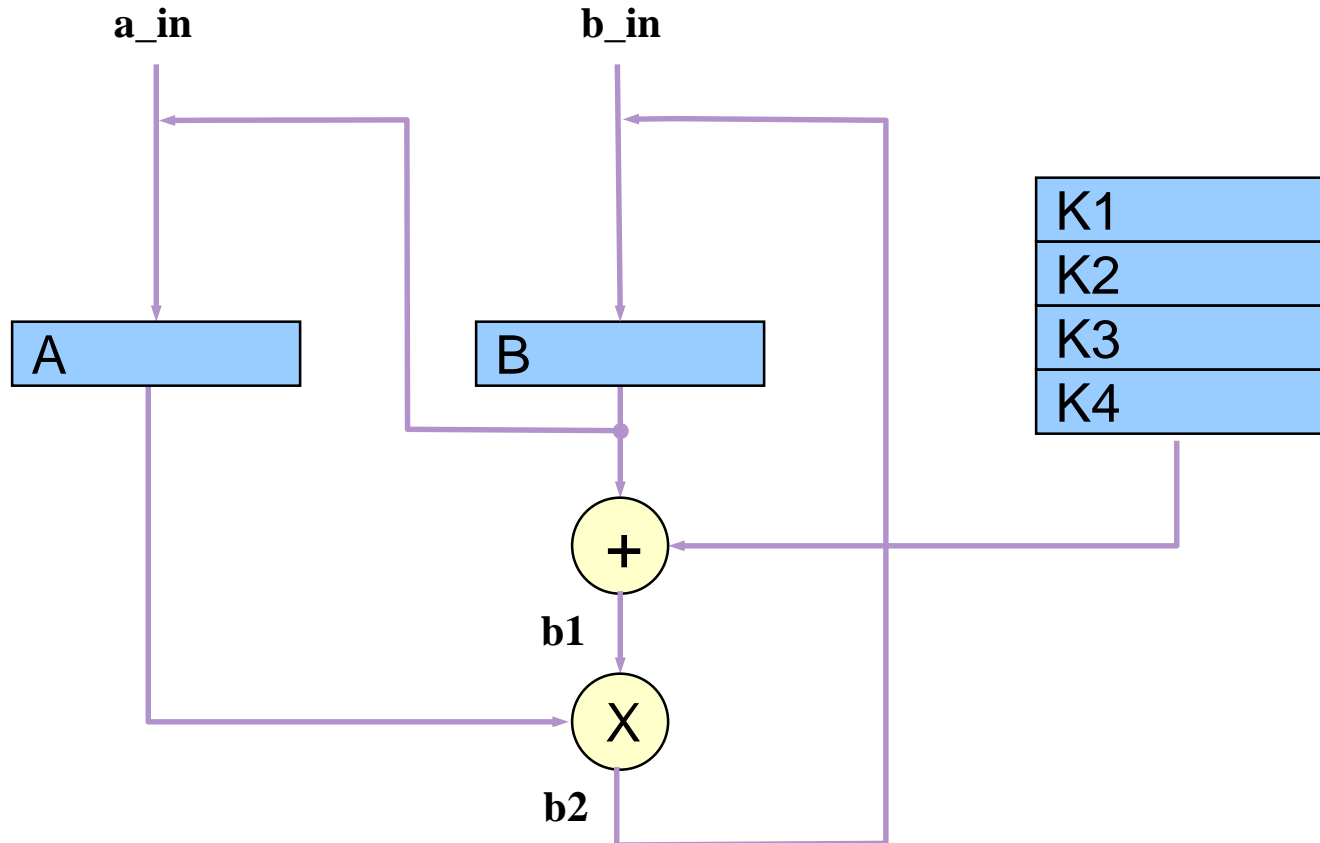
2.4: sekvenčni procesor



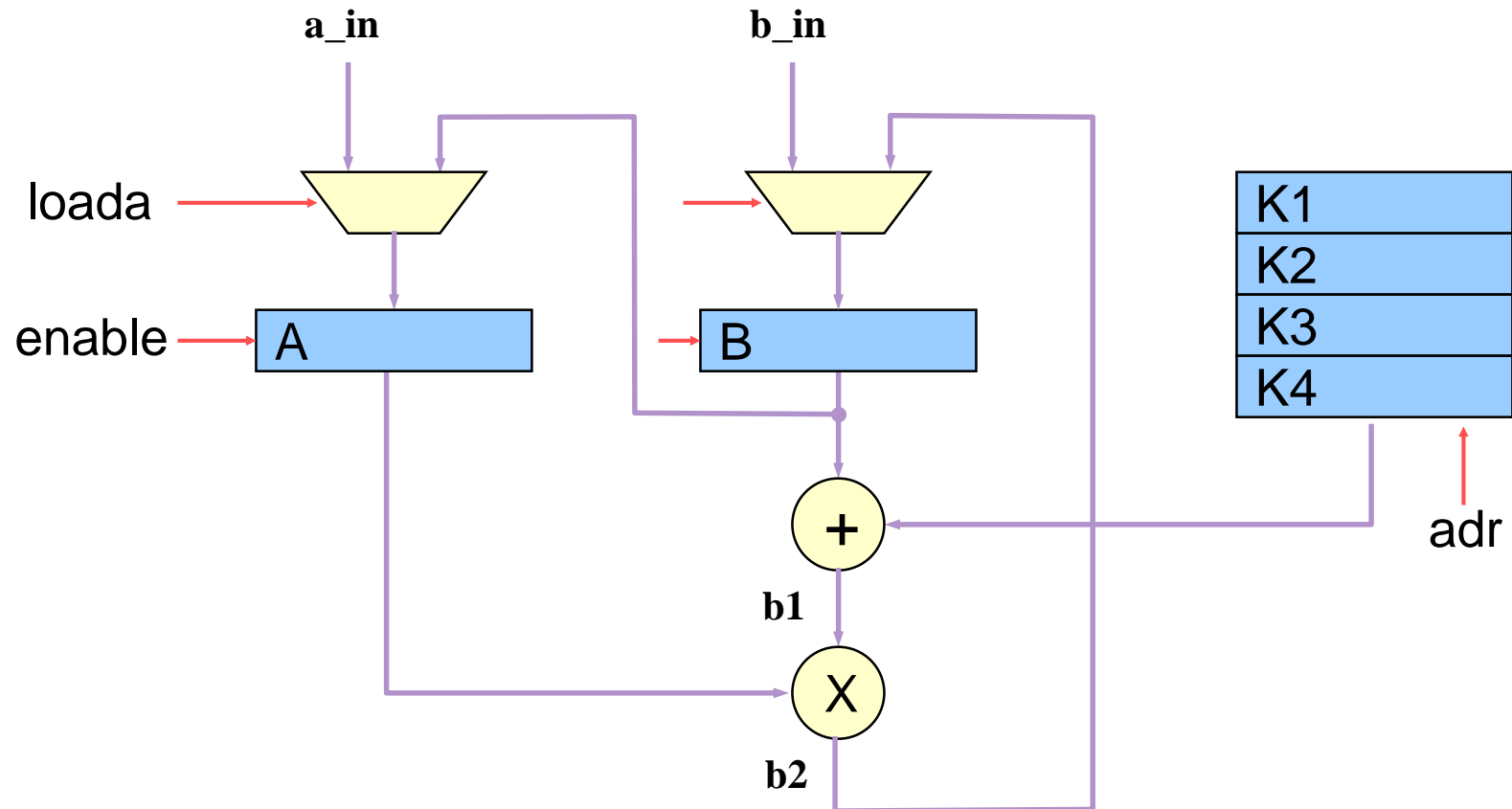
- ▶ Vezje razdelimo na podatkovni in kontrolni del



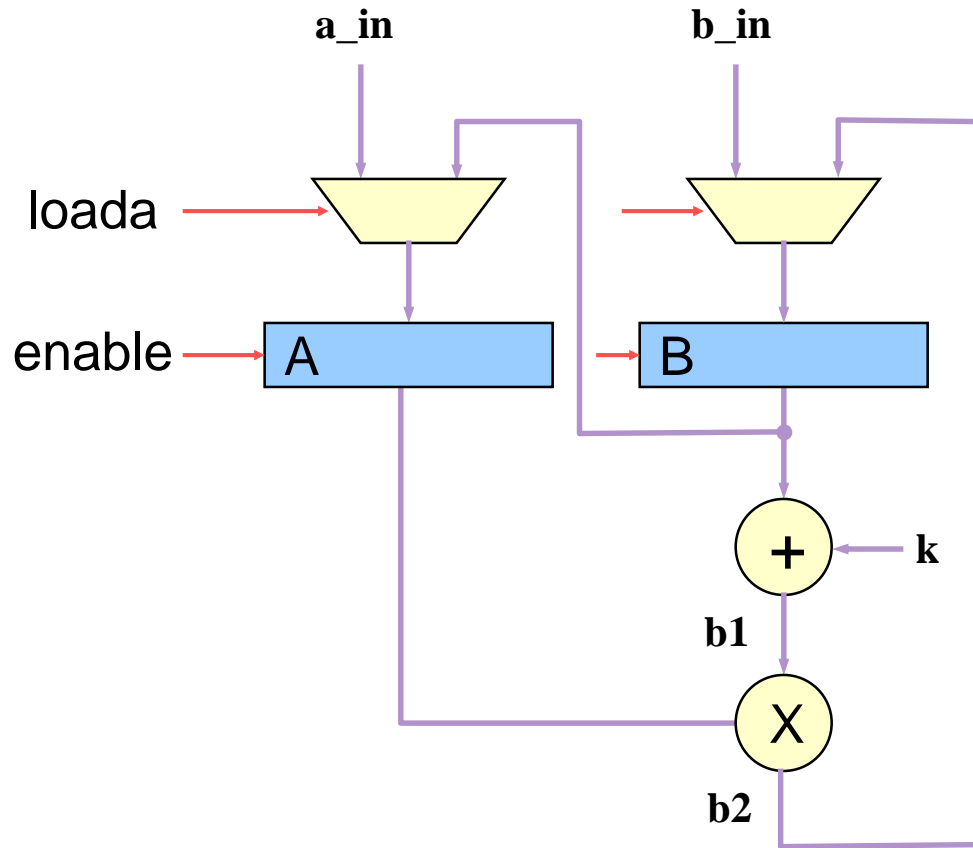
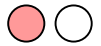
2.4: podatkovna pot



2.4: krmilni signali



2.4: 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;
```

2.4 ROM v jeziku VHDL

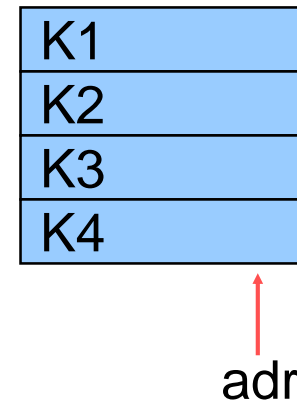


- ▶ za opis uporabimo zbirko (array)

```
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");

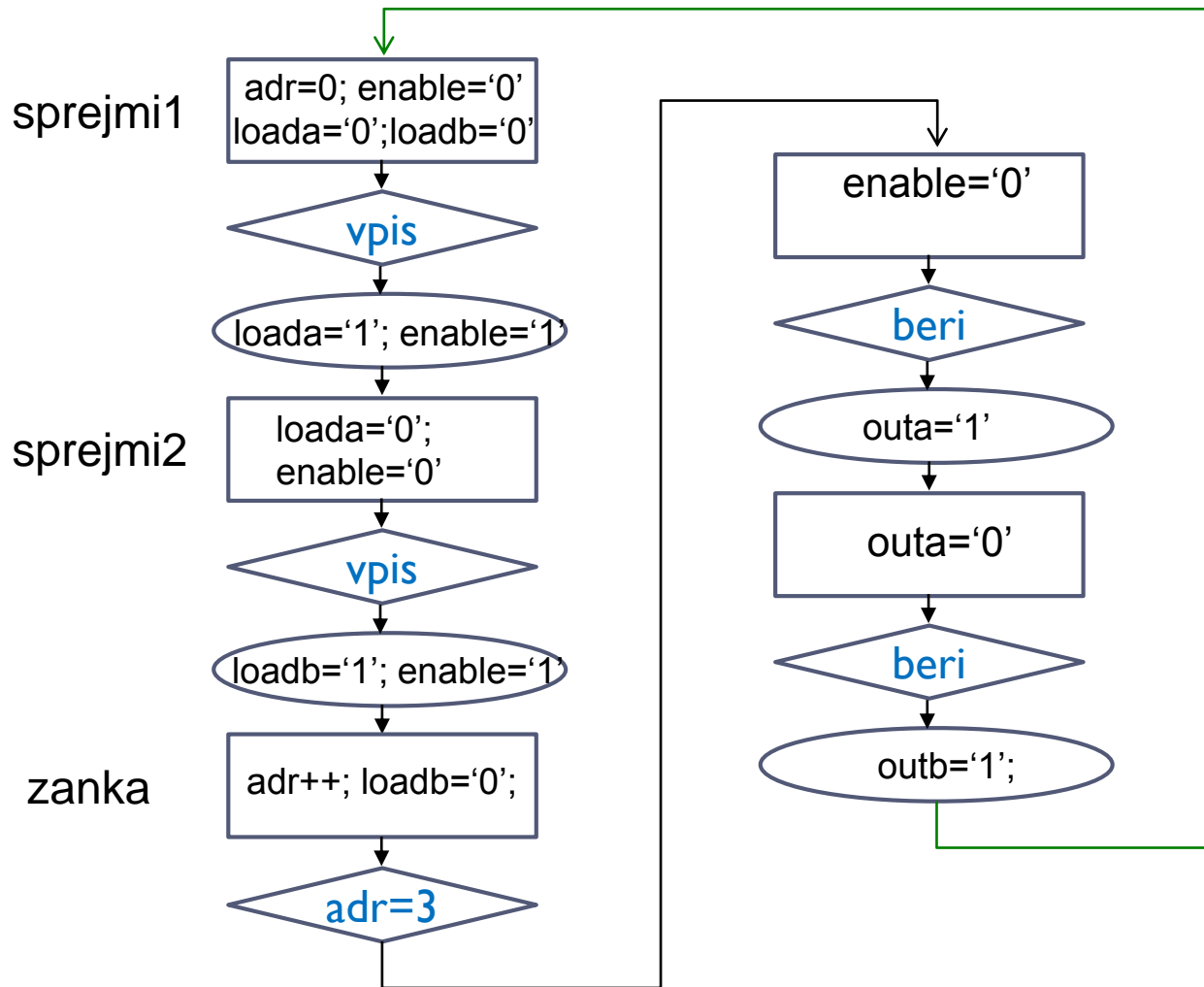
signal adr: integer range 0 to 3;
...
begin
    k <= rom(adr);
```



2.4: krmilno vezje



► Algoritmični sekvenčni avtomat



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