



Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*



Digitalni Elektronski Sistemi

Osnove jezika VHDL

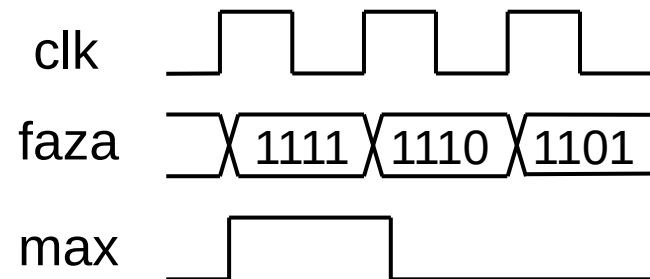
3. del: sekvenčna vezja

Pravila za modeliranje sekvenčnih vezij

- ▶ Sekvenčni del z izhodnimi flip-flopi opišemo s sinhronim procesom Register
- ▶ Kombinacijsko vezje opišemo s kombinacijskim procesom ali izven procesnega okolja Transfer Level

```
acc: process (clk)
begin
  if rising_edge(clk) then
    faza <= faza + inc;
  end if;
end process;

max <= '1' when faza="1111"
      else '0';
```



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 flip-flop(e)
 - ▶ signali niso primerni za izračun vmesnih rezultatov s kombinacijsko logiko

Spremenljivke (variable)

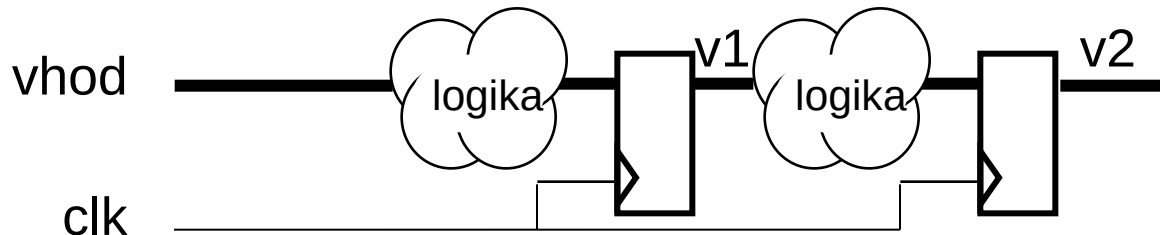
- ▶ Spremenljivke uporabljamo v procesnem okolju za izračun vmesnih rezultatov
 - ▶ ob izračunu spremenljivka **takoj** dobi novo vrednost
- ▶ Deklaracija spremenljivke:

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

- ▶ Prireditveni operator (:=)

```
v := vhod - "0011";
```

Sinteza vezja s spremenljivkami

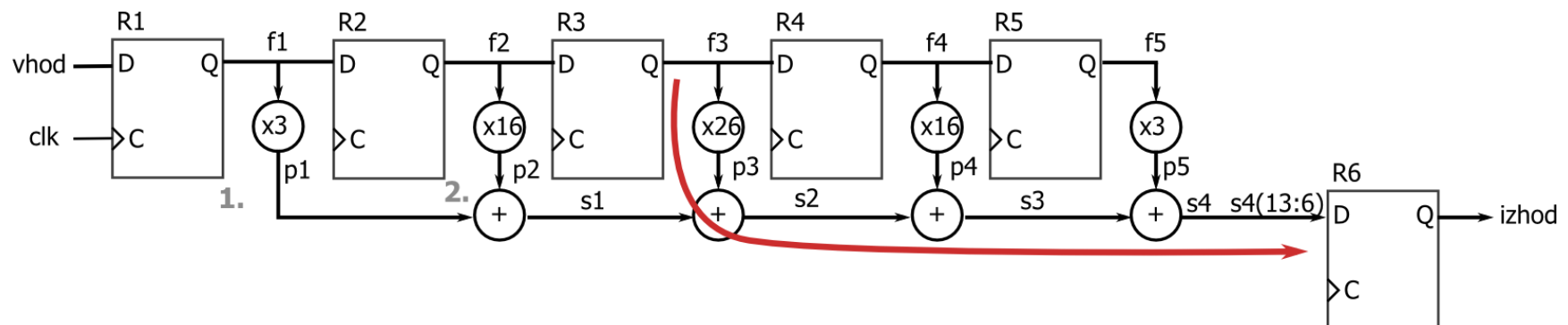


```
pipe: process (clk)
  variable v: signed (3 downto 0);
begin
  if rising_edge(clk) then
    v := vhod - "0011";
    if v > 10 then
      v1 <= v;
    else
      v1 <= "1010";
    end if;
    v2 <= v1 xor "1010";
  end if;
end process;
```

- ▶ prireditve spremenljivki ob fronti ure ni nujno izhod FF !
- ▶ FF dobimo šele po prireditvi spremenljivke signalu v1

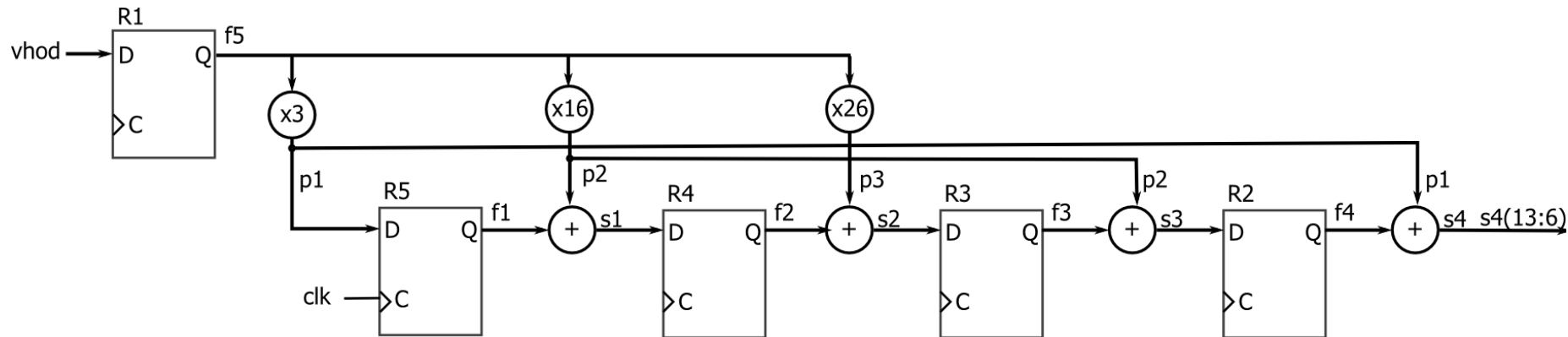
Sinteza vezja za obdelavo podatkov

- ▶ Zmogljivost vezja omejujejo zakasnitve v kombinacijskem delu: prenos pri seštevanju, ...
 - ▶ frekvenco ure omejuje povezava z največjo zakasnitvijo, ki se imenuje **kritična pot**
- ▶ Primer: Gaussovo sito
 - ▶ množenje z $26 = 11010$ in 3 seštevanja



Optimizacija sekvenčnega vezja

- ▶ Dodamo flip-flope med kombinacijske gradnike (cegovod)
- ▶ Preuredimo operacije, da zmanjšamo kritično pot
- ▶ Optimizirano Gaussovo sito
 - ▶ množenje z 26 in eno seštevanje na kritični poti



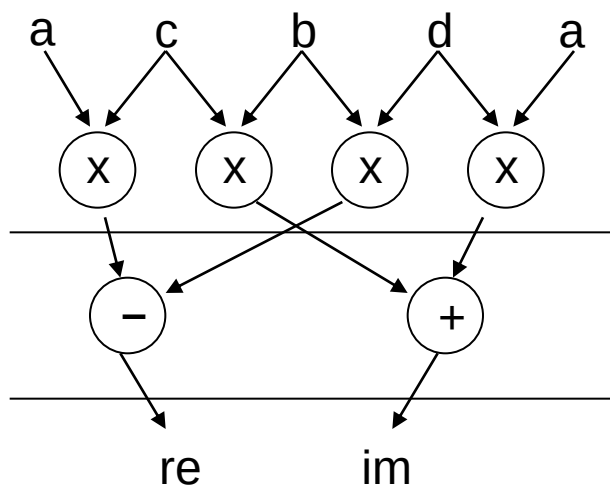
Zakasnitve odvisne od tehnologije

- ▶ Integrirana vezja ASIC: prevladujejo zakasnitve v logiki
 - ▶ zmanjšamo s topologijo vrat, obremenitvijo in načrtovanjem transistorjev
- ▶ Programirljiva vezja FPGA: 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)

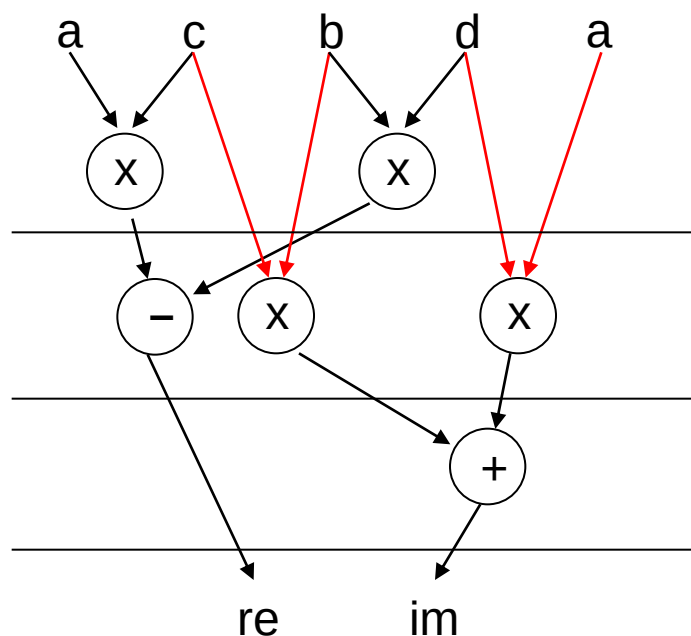
Optimizacija površine vezja

► Primer: Kompleksno množenje

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



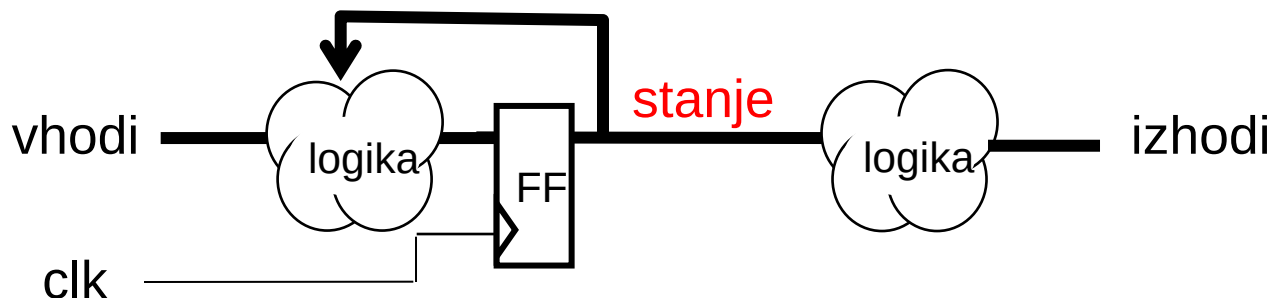
- 4 mult, 2 addsub
- 2 računski cikla



- 2 mult, 1 addsub
- 3 računski cikla

Stroj s končnim številom stanj (avtomat)

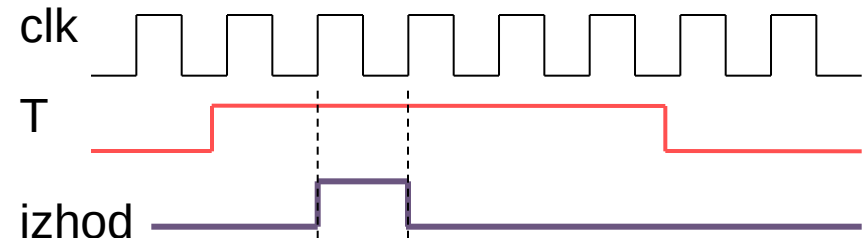
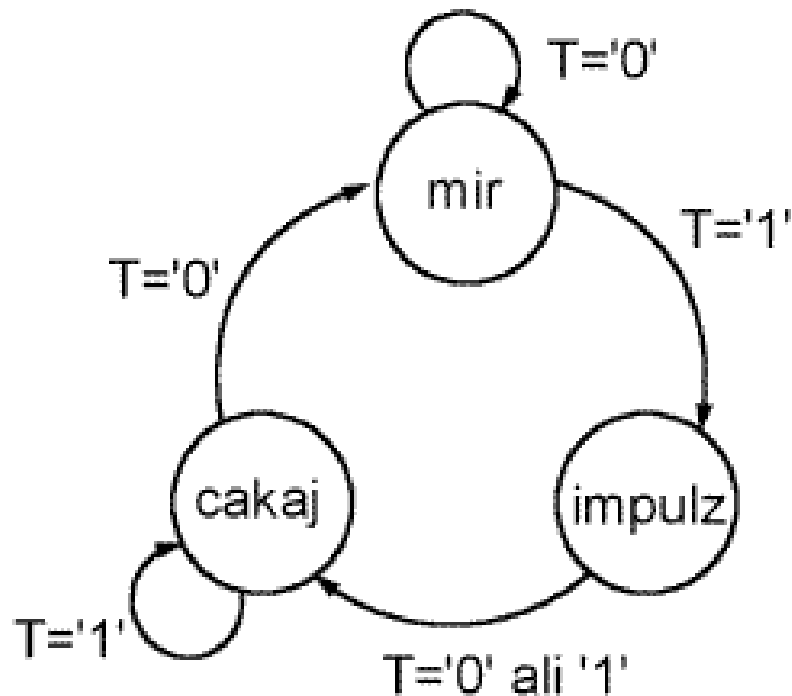
- ▶ Stroj stanj je abstrakcija iz teoretičnega računalništva
- ▶ Sekvenčno vezje, ki deluje na principu **stanja** in vsebuje
 - ▶ kombinacijsko logiko
 - ▶ flip-flope za shranjevanje stanja
- ▶ Npr: Moorov avtomat



- ▶ Ob fronti ure kombinacijska logika določa izhode in naslednje stanje, ki je odvisno od vhodov in trenutnega stanja

Diagram prehajanja stanj

- ▶ Primer: vezje za detekcijo pritiska tipke
 - ▶ na izhodu naredi en impulz ob pritisku na tipko ($T='1'$)



Opis Moorovega avtomata v jeziku VHDL

- ▶ Določimo podatkovni tip, kjer naštejemo stanja
 - ▶ sinhroni proces za shranjevanje stanj

```
architecture RTL of avtomat is  
  type stanja is (mir, impulz, cakaj);  
  signal stanje, naslednje: stanja;  
begin
```

```
  FF: process (clk)  
  begin  
    if rising_edge(clk) then  
      stanje <= naslednje;  
    end if;  
  end process;
```

prehodi med stanji:

```
  preh: process (stanje, T)  
  begin  
    case stanje is  
      when mir =>  
        if T = '1' then  
          naslednje <= impulz;  
        else  
          naslednje <= mir;  
        end if;  
      when impulz =>  
        naslednje <= cakaj;
```

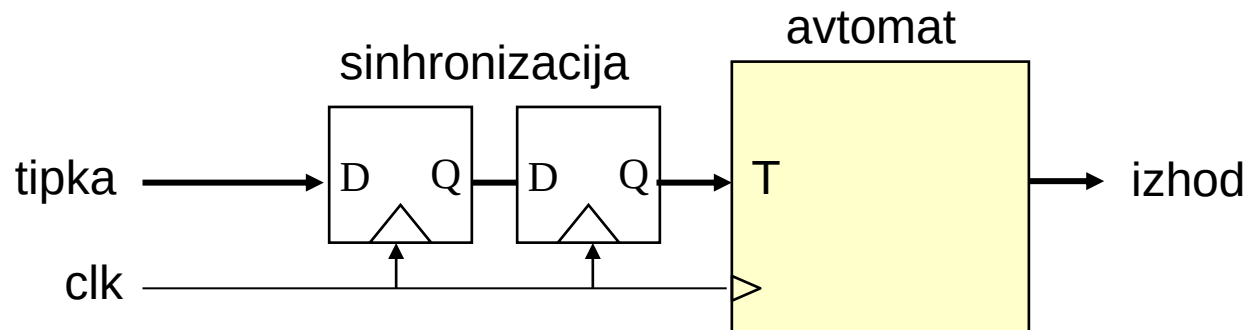
Izhodna logika

► Kombinacijska logika za izhode

- kombinacijski proces ali izbirni stavek when ... else

```
komb: process (stanje)
begin
  if stanje=impulz then
    izhod <= '1';
  else
    izhod <= '0';
  end if;
end process;
```

- Vhod v avtomat je potrebno sinhronizirati !



Sinteza in kodiranje stanj

- ▶ Kako so kodirana stanja?
 - ▶ kodiranje stanj naredi program za sintezo vezja
 - ▶ nekatere kode ne predstavljajo stanja iz diagrama
 - ▶ V stavku **case** uporabimo **when others=>** kjer določimo v katero stanje naj se premakne avtomat, če pride slučajno v nedefinirano stanje

binarno

mir = 00

impulz = 01

cakaj = 10

one-hot

mir = 001

impulz = 010

cakaj = 100

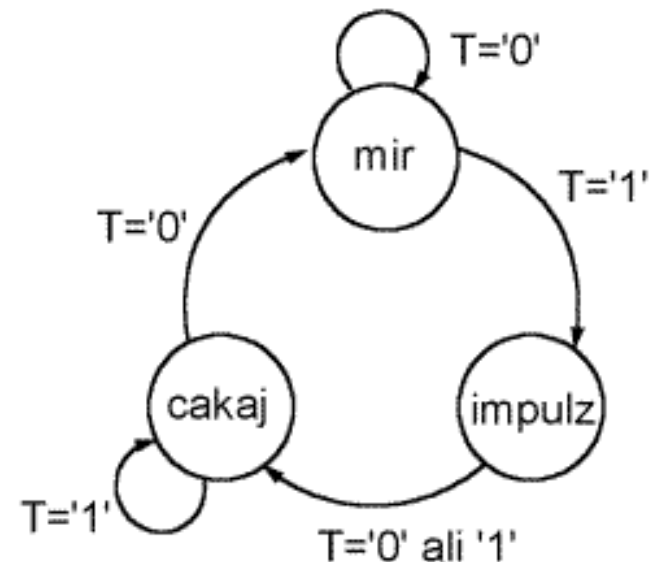
Kompakten opis v jeziku VHDL

- ▶ Kompakten opis sekvenčnega stroja stanj
 - ▶ v enem procesu opišemo flip-flope in vhodno logiko

```
sekv: process (clk)
begin
  if rising_edge(clk) then
    case stanje is
      when mir =>
        if tipka='1' then
          stanje <= impulz;
        end if;

        when impulz =>
          stanje <= cakaj;

        when others =>
          if tipka='0' then
            stanje <= mir;
          end if;
        end case;
```

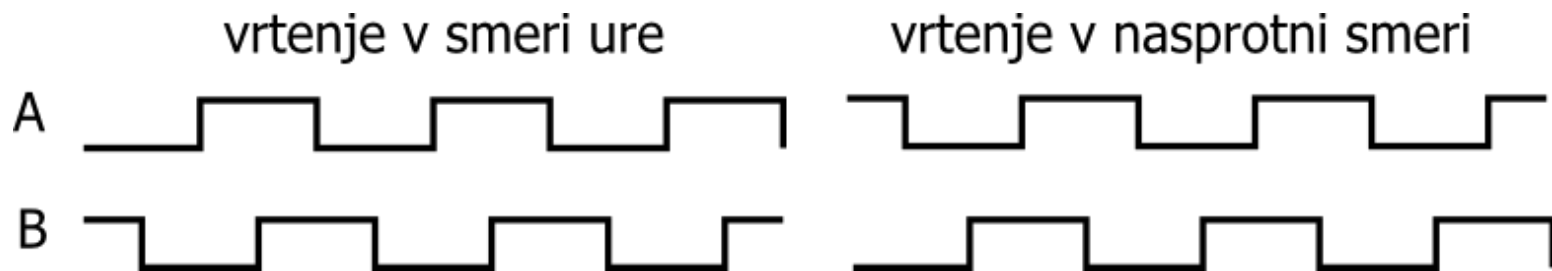


```
izhod <= '1' when stanje=impulz
           else '0';
```

Rotacijski kodirnik

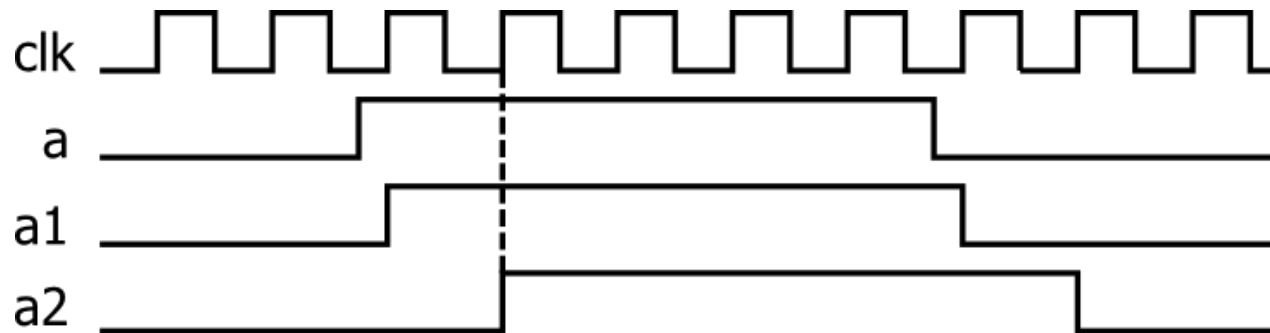


- ▶ kodirnik ima dva izhoda, ki ob vrtenju generirata impulze

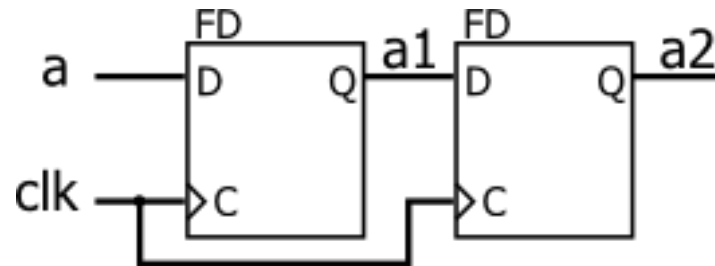


Sinhrona detekcija fronte signala

- Kako zaznati fronto asinhronnega vhoda ?



```
vz: process (clk)
begin
  if rising_edge(clk) then
    a1 <= a;
    a2 <= a1;
  end if;
end process;
```



```
a_rise <= '1' when a1='1' and a2='0' else '0';
```

Sekvenčno vezje za detekcijo vrtenja

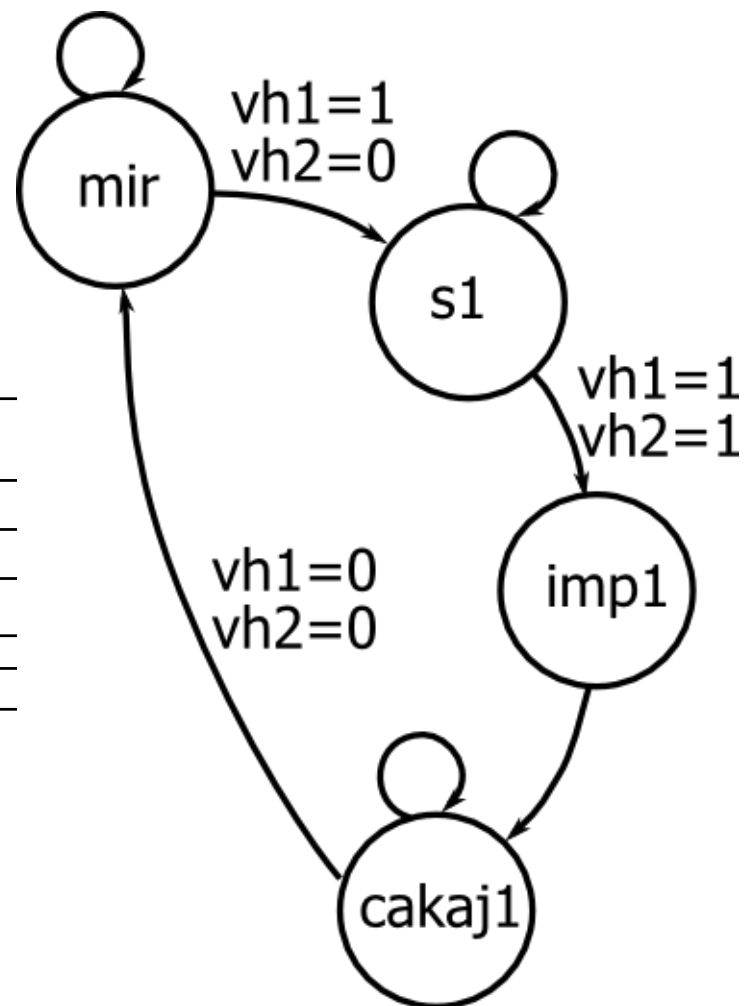
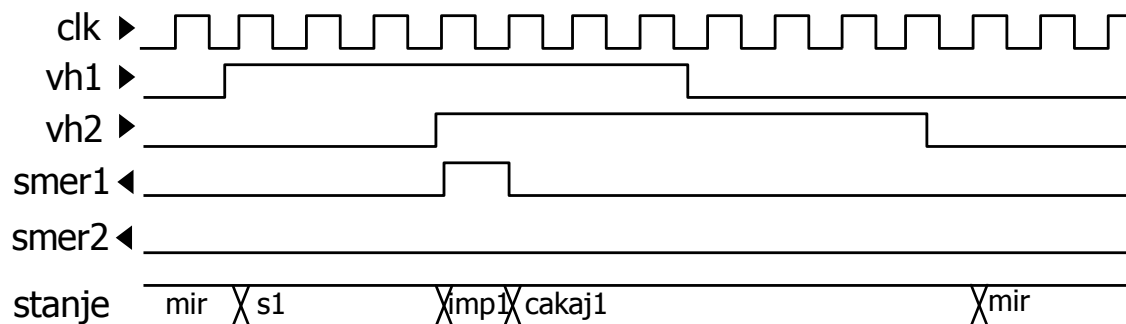
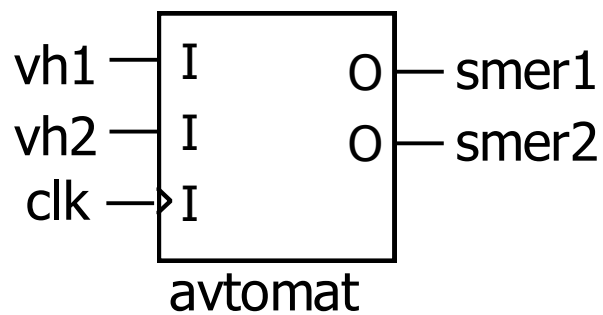
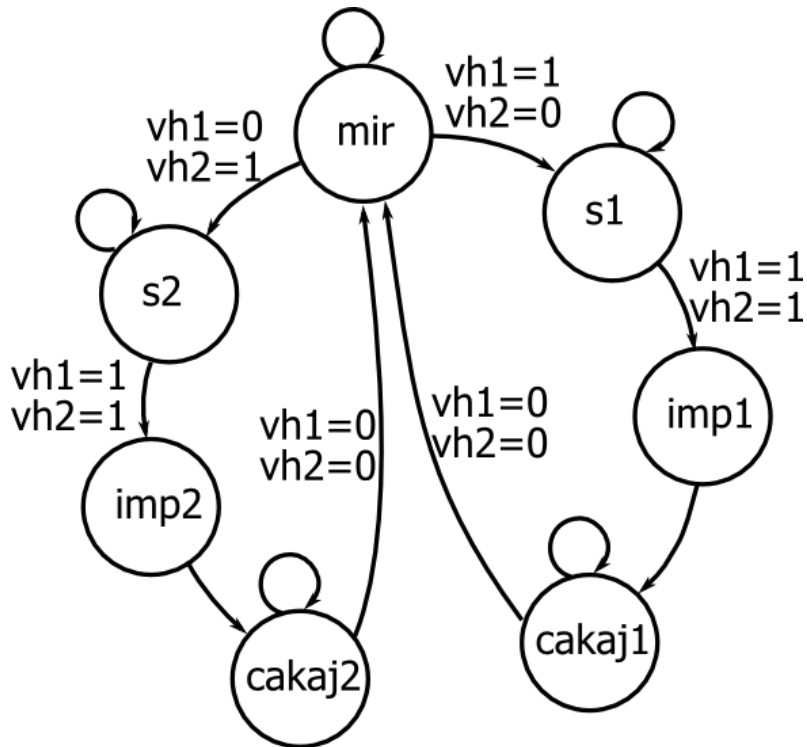


Diagram stanja



architecture RTL of avtomat is

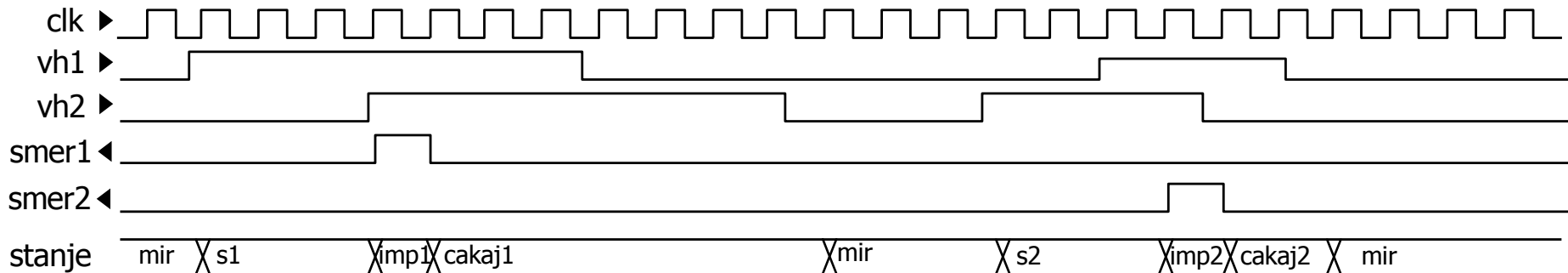
type stanja **is** (mir, s1, imp1, s2, imp2, cakaj);

signal st: stanja;

begin

smer1 <= '1' **when** st=imp1 **else** '0';

smer2 <= '1' **when** st=imp2 **else** '0';



Opis prehajanja stanj

```
avt: process (clk)
begin
  if rising_edge(clk) then
    case st is
      when mir =>
        if vh1='1' and vh2= '0' then
          st <= s1;
        elsif vh1='0' and vh2= '1' then
          st <= s2;
        end if;

      when s1 =>
        if vh1='1' and vh2= '1' then
          st <= imp1;
        end if;
```

```
      when imp1 =>
        st <= cakaj;

      when s2 =>
        if vh1='1' and vh2= '1' then
          st <= imp2;
        end if;

      when imp2 =>
        st <= cakaj;

      when others =>
        if vh1='0' and vh2='0' then
          st <= mir;
        end if;
      end case;
    end if;
  end process;
```