Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

Digitalni Elektronski Sistemi

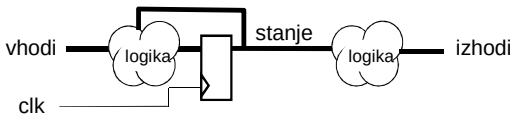
Osnove jezika VHDL

4. del: sekvenčna vezja s povratno zanko

1

Končni avtomat (Finite State Machine)

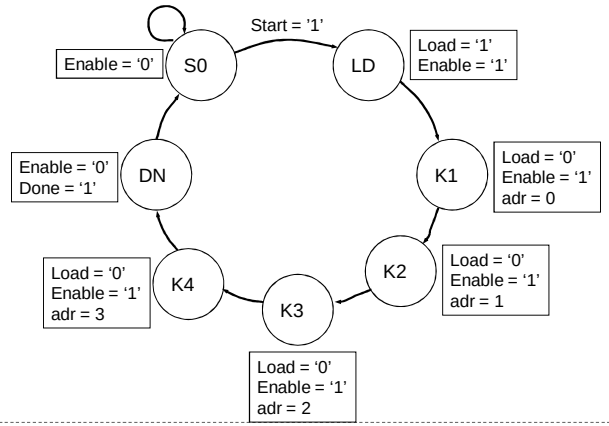
- Sekvenčno vezje s povratno zanko
  - register stanj
  - vhodna logika
  - Izhodna logika
- Moorov avtomat:



2

Primer končnega avtomata

- Pri vsakem stanju določimo vrednosti izhodov



3

Opis avtomata v jeziku VHDL

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

```
architecture RTL of avtomat is
 ...
 type stanja is (s0, ld, k1, k2, k3, k4, dn);
 signal stanje, naslednje: stanja;
...
begin

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

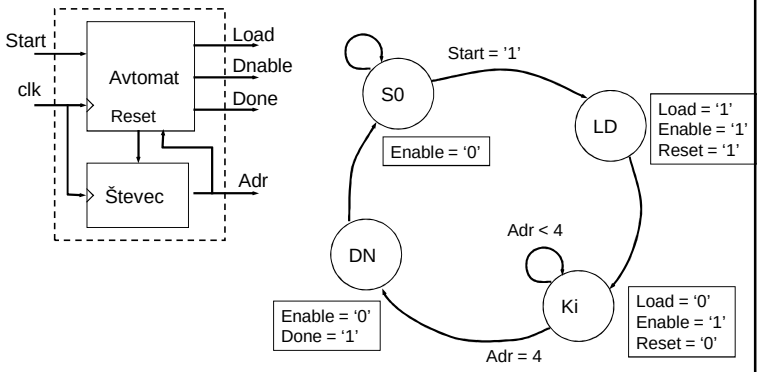
prehodi med stanji:

```
preh: process (stanje, start, adr)
begin
 case stanje is

  when s0 =>
   if start = '1' then
    naslednje <= ld;
   else
    naslednje <= s0;
   end if;

  when ld =>
    naslednje <= k1;
```

4

Avtomat z zunanjim števcem

- namesto stanj v katerih spreminjamo adr uporabimo zunanji števec



5

Izhodna logika in števec

- Kombinacijska logika za izhode (when … else)
- Sinhroni proces za zunanji števec

```
Load <= '1' when stanje=ld else '0';
Enable <= '0' when stanje=s0 or stanje=dn else '1';
Done <= '1' when stanje=dn else '0';
Reset <= '1' when stanje=ld else '0';

stev: process(clk)
begin
  if rising_edge(clk) then
    if Reset='1' then          -- mora biti sinhroni
     Adr <= 0;
    else
     Adr <= Adr + 1;
    end if;
  end if;
end process;
```

- kombinacijski izhodi imajo šum ob prehajanju stanj
- ne smemo jih uporabiti za uro ali asinhroni reset…
- šum odstranimo, če jih pošjemo čez D flip-flop

6

Vhodni signali

- Vhodni signali v sinhroni avtomat morajo biti sinhronizirani z uro !
  - sicer prekršimo dinamični red in avtomat gre v poljubno stanje
- 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
S0= 000
LD= 001
K1= 010
K2= 011
K3= 100
K4= 101
DN=110

one-hot
S0 = 0000001
LD = 0000010
K1= 0000100
K2= 0001000
K3= 0010000
K4= 0100000
DN= 1000000

7

Končni avtomat z registrskimi operacijami

- Kompakten opis sekvenčnega vezja
  - V enem procesu, opišemo register in prehajanje med stanji ter operacije (npr. nalaganje registra, povečevanje – števec…)

```
avtomat: process (clk)
begin
  if rising_edge(clk) then
    case stanje is

    when s0 =>
      if start='1' then
        stanje <= ld;
        reg <= vhod;
      end if;

    when ld =>
      stanje <= ki;
      Adr <= 0;

    when ki =>
      if Adr<3 then
        Adr <= Adr + 1;
      else
        stanje <= dn;
      end if;

    when others =>
      stanje <= s0;
    end case;
  end if;
end process;
```

8