11. Sekvencijski sklopovi - primjer

Primjer 8.6 iz knjige:
S. D. Brown, Z. G. Vranešić: Fundamentals of Digital Logic with VHDL Design, McGraw-Hill, 2001,
str. 475-480.

Zadatak - neformalna specifikacija

- Automat prihvaća kovanice:
  - *nickel*, 5 centi (5/100 $) i
  - *dime*, 10 centi (10/100 $).

- Kad se ubaci ukupno 15 centi automat izbacuje traženi artikl.

- Ako se ubaci 20 automat ne vraća ostatak već "kreditira" kupca s 5 u sljedećoj narudžbi.

Pretpostavke

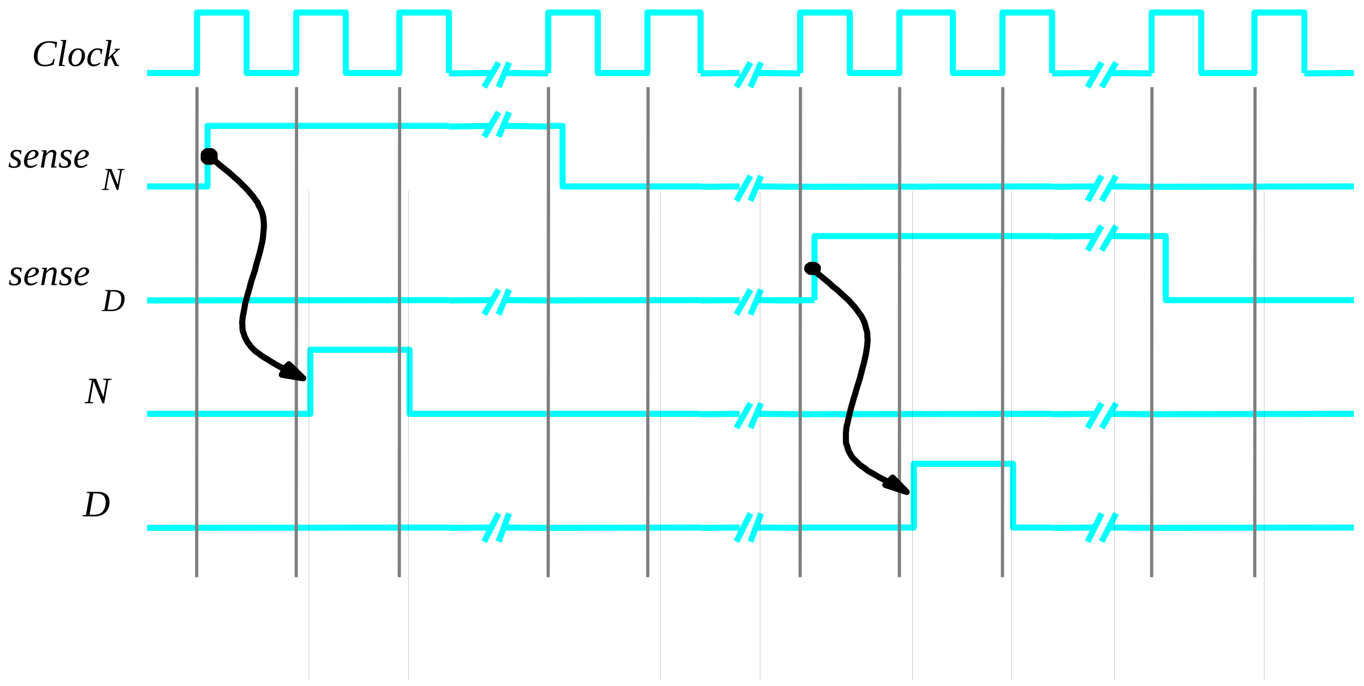
- Sinkroni rad
- Uzlaznim bridom (pozitivnim) okidani bistabili
- Takt impulsi reda veličine 100 ns

- Mehanizam koji prihvaća kovanice generira dva signala čije prisustvo označava koji je novčić ubačen:
  - $sense_D$
  - $sense_N$

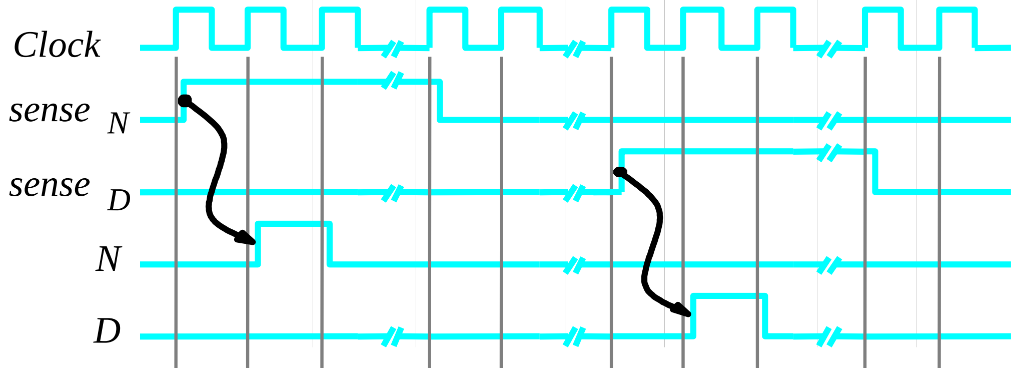
- Kako je taj mehanizam mehanički, pretpostavljamo da je puno sporiji od “elektronike” te će signali biti prisutni veći broj takt impulsa.

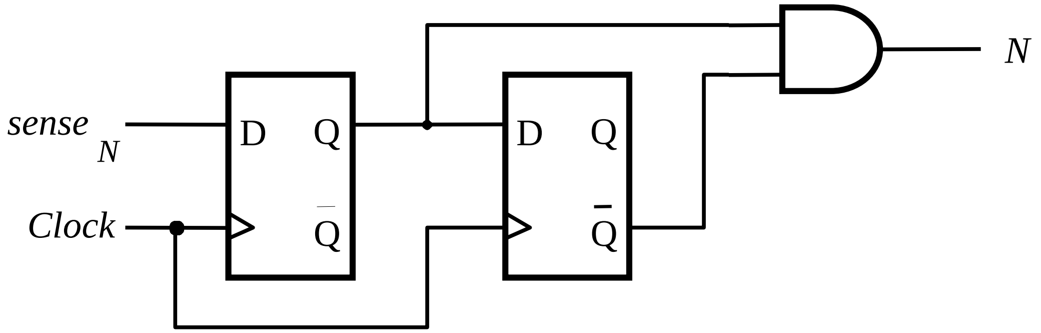
Pretpostavke

Automat ("mehanički dio") generira i dva signala: *D* i *N*. *D* je postavljen u "1" u trajanju 1 takt impulsa nakon što D_S postane "1", a *N* je postavljen u "1" u trajanju 1 takt impulsa nakon što N_S postane "1"

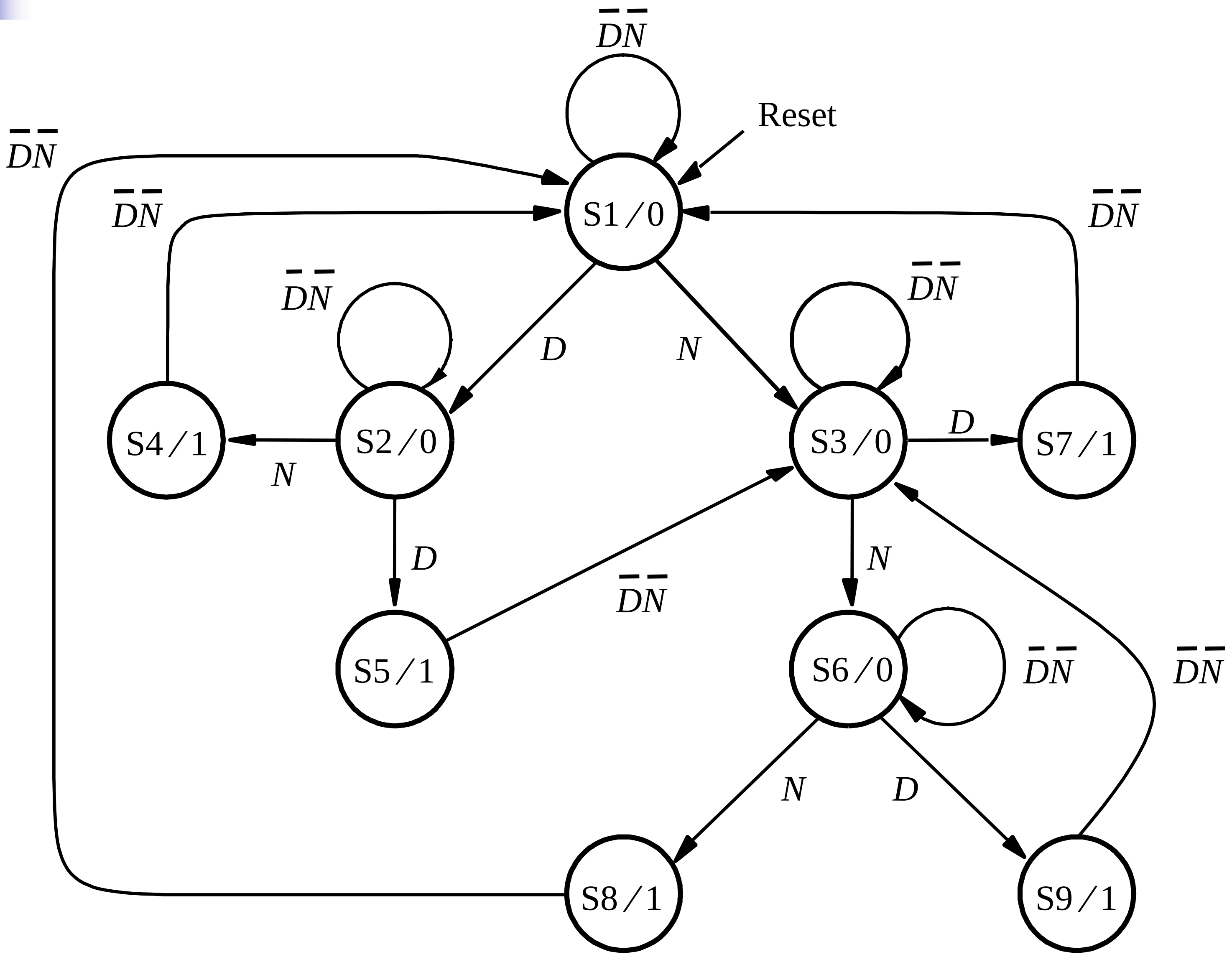


Sklop za generiranje signala N





Dijagram stanja



Tablica stanja

Trenutno stanje	Sljedeće stanje				Izlaz z
	DN = 00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Ekvivalentna stanja

Trenutno stanje	Sljedeće stanje				Izlaz z
	DN =00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

P_1 = (S1,S2,S3,S4,S5,S6,S7,S8,S9)

S obzirom na izlaze:

P_2 = (S1,S2,S3,S6) (S4,S5,S7,S8,S9)

Ekvivalentna stanja

Trenutno stanje	Sljedeće stanje				Izlaz z
	DN =00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Sad gledamo slijedeća stanja:

P_2 = (S1,S2,S3,S6) (S4,S5,S7,S8,S9)

DN 00	S1,S2,S3,S6	S1,S3,S1,S1,S3
DN 01	S3,S4,S6,S8	- , - , - , - , -
DN 10	S2,S5,S7,S9	- , - , - , - , -

Slijedi nova podjela:
P_3 = (S1) (S3) (S2,S6) (S4,S5,S7,S8,S9)

Ponovo gledamo slijedeća stanja:

P_3 = (S1) (S3) (S2,S6) (S4,S5,S7,S8,S9)

DN 00	S1	S3	S2,S6	S1,S3,S1,S1,S3
DN 01	S3	S6	S4,S8	- , - , - , - , -
DN 10	S2	S7	S5,S9	- , - , - , - , -

Nova podjela:
P_4 = (S1) (S3) (S2,S6) (S4,S7,S8) (S5,S9)

Ekvivalentna stanja

Trenutno stanje	Sljedeće stanje				Izlaz
	DN =00	01	10	11	*z*
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Ponovo gledamo sljedeća stanja:

	P_4 = (S1)	(S3)	(S2,S6)	(S4,S7,S8)	(S5,S9)
DN 00			S2,S6	S1,S1,S1	S3,S3
DN 01			S4,S8	- , - , -	- , -
DN 10			S5,S9	- , - , -	- , -

Konačna podjela:

$$\begin{array}{lccccc} P_5 = & (S1) & (S3) & (S2,S6) & (S4,S7,S8) & (S5,S9) \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & S1 & S3 & S2 & S4 & S5 \end{array}$$

Ekvivalentna stanja – nova tablica stanja

Trenutno stanje	Sljedeće stanje				Izlaz
	DN= 00	01	10	11	z
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Konačna podjela:

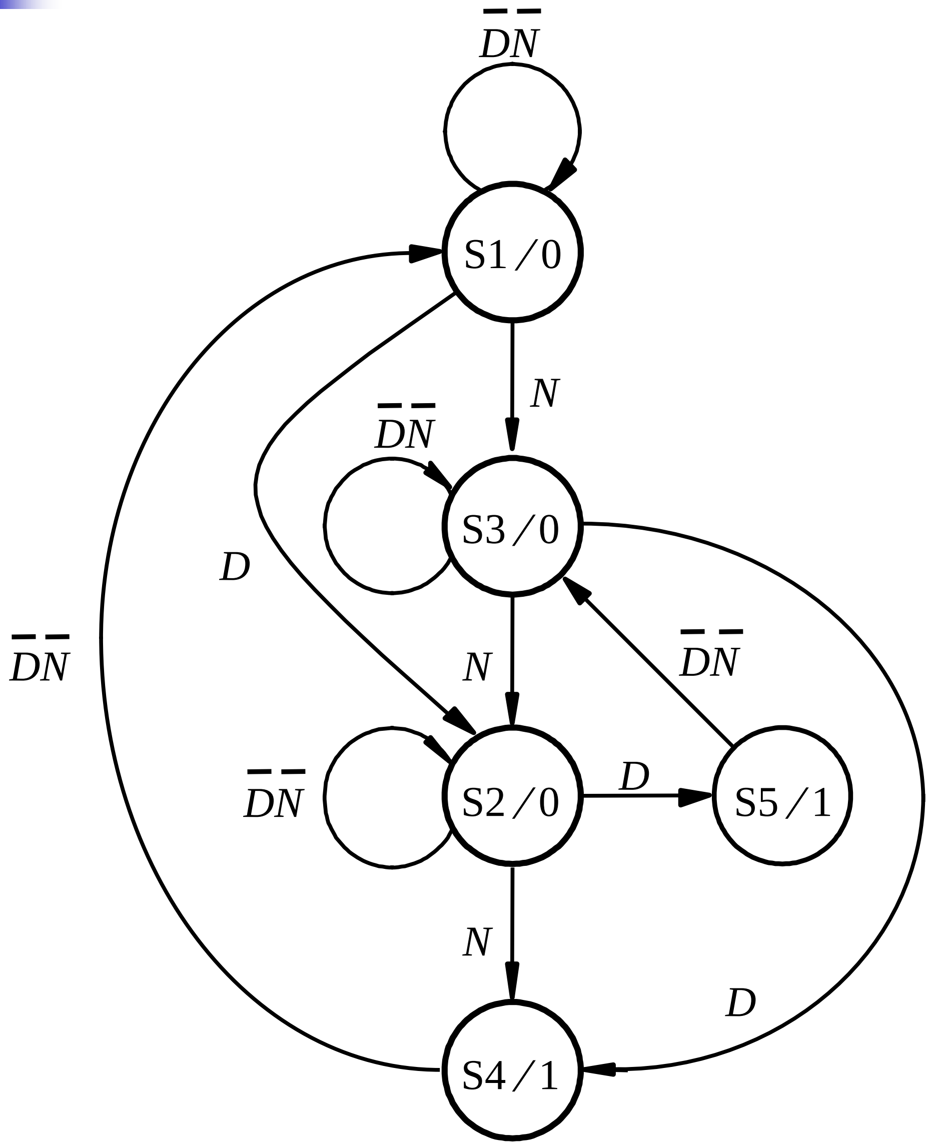
(S1) (S3) (S2,S6) (S4,S7,S8) (S5,S9)

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

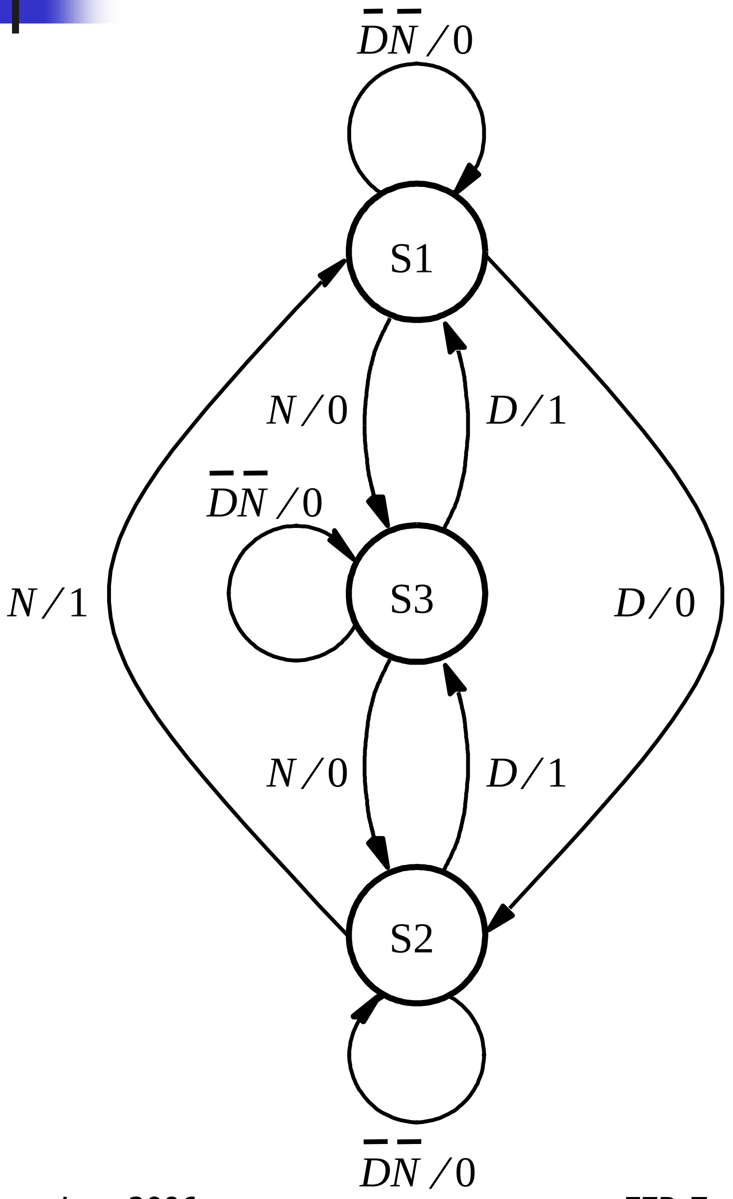
S1 S3 S2 S4 S5

Trenutno stanje	Sljedeće stanje				Izlaz
	DN= 00	01	10	11	z
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

Novi dijagram stanja – minimalni automat



Mealyjev model automata



VHDL – “state machine”

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity machine is
    Port ( D, N : in  STD_LOGIC;
              CP : in STD_LOGIC;
              RESET : in STD_LOGIC;
              Z : out  STD_LOGIC);
end machine;

architecture Behavioral of machine is
   type state_type is (S1,S2,S3,S4,S5);
   signal state, next_state : state_type;
   signal Z_int : std_logic;
   signal DN : std_logic_vector (0 to 1);
begin
   DN <= D & N;
   SYNC_PROC: process (CP)
   …
   OUTPUT_DECODE: process (state)
   …
   NEXT_STATE_DECODE: process (state, DN)
   …
end Behavioral;
```

VHDL

```
SYNC_PROC: process (CP)
  begin
    if (CP'event and CP = '1') then
      if (RESET = '1') then
        state <= S1;
        Z <= '0';
      else
        state <= next_state;
        Z <= Z_int;
      end if;
    end if;
  end process;

OUTPUT_DECODE: process (state)      -- MOORE State Machine - Outputs based on state only
  begin
    if    state = S4 then Z_int <= '1';
    elsif state = S5 then Z_int <= '1';
    else                  Z_int <= '0';
    end if;
  end process;
```

VHDL

```
NEXT_STATE_DECODE: process (state, DN)
begin
   --declare default state for next_state to avoid latches
   next_state <= state;  --default is to stay in current state
   --insert statements to decode next_state
   case (state) is
      when S1 =>
         case (DN) is
               when "00" => next_state <= S1;
               when "01" => next_state <= S3;
               when "10" => next_state <= S2;
               when others => next_state <= state;
         end case;
      when S2 =>
        case (DN) is
               when "00" => next_state <= S2;
               when "01" => next_state <= S4;
               when "10" => next_state <= S5;
               when others => next_state <= state;
         end case;
         . . . .
```

VHDL

```
        when S3 =>
          case (DN) is
                when "00" => next_state <= S3;
                when "01" => next_state <= S2;
                when others => next_state <= state;
          end case;
         when S4 =>
          case (DN) is
                when "00" => next_state <= S1;
                when others => next_state <= state;
          end case;
         when S5 =>
          case (DN) is
                when "00" => next_state <= S3;
                when others => next_state <= state;
          end case;
         when others =>
          next_state <= state;
      end case;
    end process;

  end Behavioral;
```

Kodiranje stanja

Trenutno stanje	Sljedeće stanje $DN = 00$	01	10	11	Izlaz z
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

ABC	Sljedeće stanje $DN = 00$	01	10	11	Izlaz z
S1…000	000	001	010	–	0
S2…010	010	011	100	–	0
S3…001	001	010	011	–	0
S4…011	000	–	–	–	1
S5…100	001	–	–	–	1

Ulazne jednadžbe - bistabil A

	Sljedeće stanje				Izlaz
ABC	$DN=00$	01	10	11	z
S1…000	000	001	010	–	0
S2…010	010	011	100	–	0
S3…001	001	010	011	–	0
S4…011	000	–	–	–	1
S5…100	001	–	–	–	1

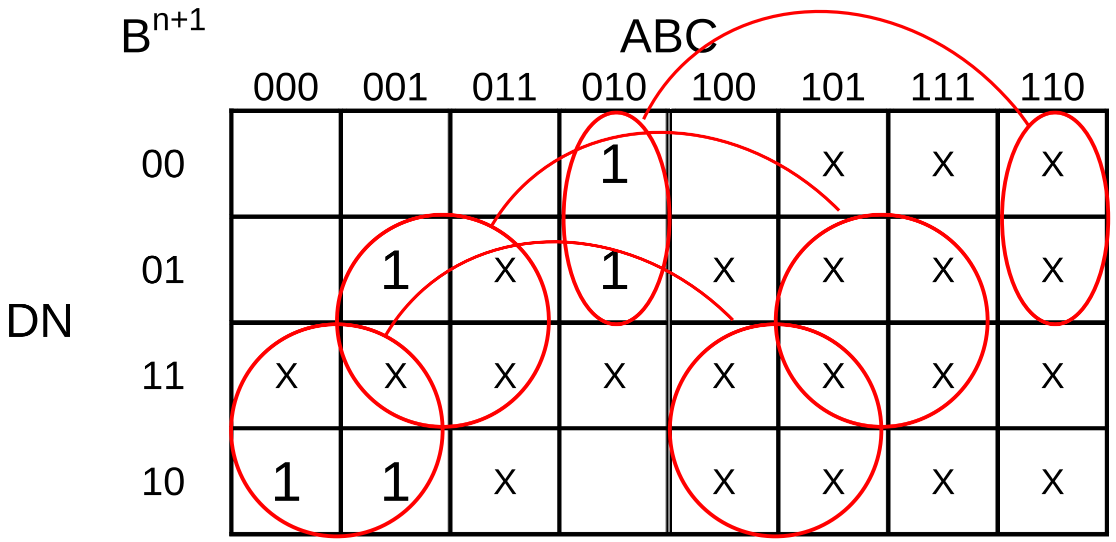
A^{n+1}

DN \ ABC	000	001	011	010	100	101	111	110
00						X	X	X
01			X		X	X	X	X
11	X	X	X	X	X	X	X	X
10			X	1	X	X	X	X

$A^{n+1}=BD$

Ulazne jednadžbe - bistabil B

	Sljedeće stanje				Izlaz
ABC	*DN* = 00	01	10	11	*z*
S1…000	000	001	010	–	0
S2…010	010	011	100	–	0
S3…001	001	010	011	–	0
S4…011	000	–	–	–	1
S5…100	001	–	–	–	1

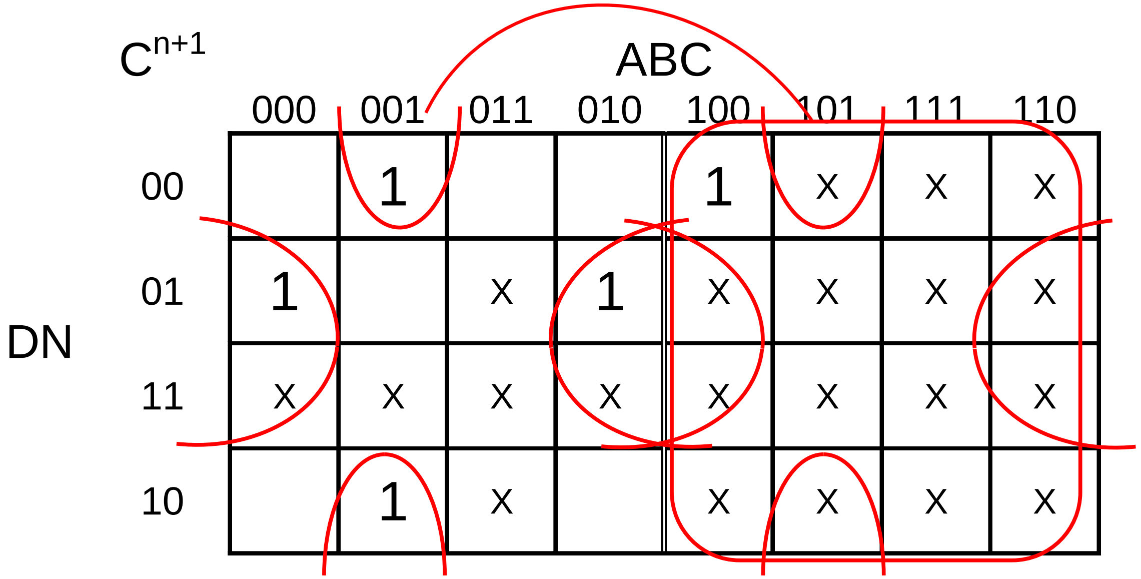


B^{n+1} DN \ ABC	000	001	011	010	100	101	111	110
00				1		X	X	X
01		1	X	1	X	X	X	X
11	X	X	X	X	X	X	X	X
10	1	1	X		X	X	X	X

$$B^{n+1}=\overline{B}D+CN+B\overline{C}\overline{D}$$

Ulazne jednadžbe - bistabil C

ABC	Sljedeće stanje *DN* = 00	01	10	11	Izlaz *z*
S1…000	000	001	010	–	0
S2…010	010	011	100	–	0
S3…001	001	010	011	–	0
S4…011	000	–	–	–	1
S5…100	001	–	–	–	1



C^{n+1} DN \ ABC	000	001	011	010	100	101	111	110
00		1			1	X	X	X
01	1		X	1	X	X	X	X
11	X	X	X	X	X	X	X	X
10		1	X		X	X	X	X

$$C^{n+1}=A+\overline{B}C\overline{N}+\overline{C}N$$

Izlazna jednadžba - izlaz Z

	Sljedeće stanje				Izlaz
ABC	*DN* = 00	01	10	11	*z*
S1…000	000	001	010	–	0
S2…010	010	011	100	–	0
S3…001	001	010	011	–	0
S4…011	000	–	–	–	1
S5…100	001	–	–	–	1

$$Z=\bar{A}BC+A\bar{B}\bar{C}$$

Z^n

DN \ ABC	000	001	011	010	100	101	111	110
00			1		1	X	X	X
01			X		X	X	X	X
11	X	X	X	X	X	X	X	X
10			X		X	X	X	X

$$Z=A+BC$$

Ulazne jednadžbe bistabila i izlazna jednadžba

$A^{n+1}=BD$

$B^{n+1}=\bar{B}D+CN+B\bar{C}\bar{D}$

$C^{n+1}=A+\bar{B}C\bar{N}+\bar{C}N$

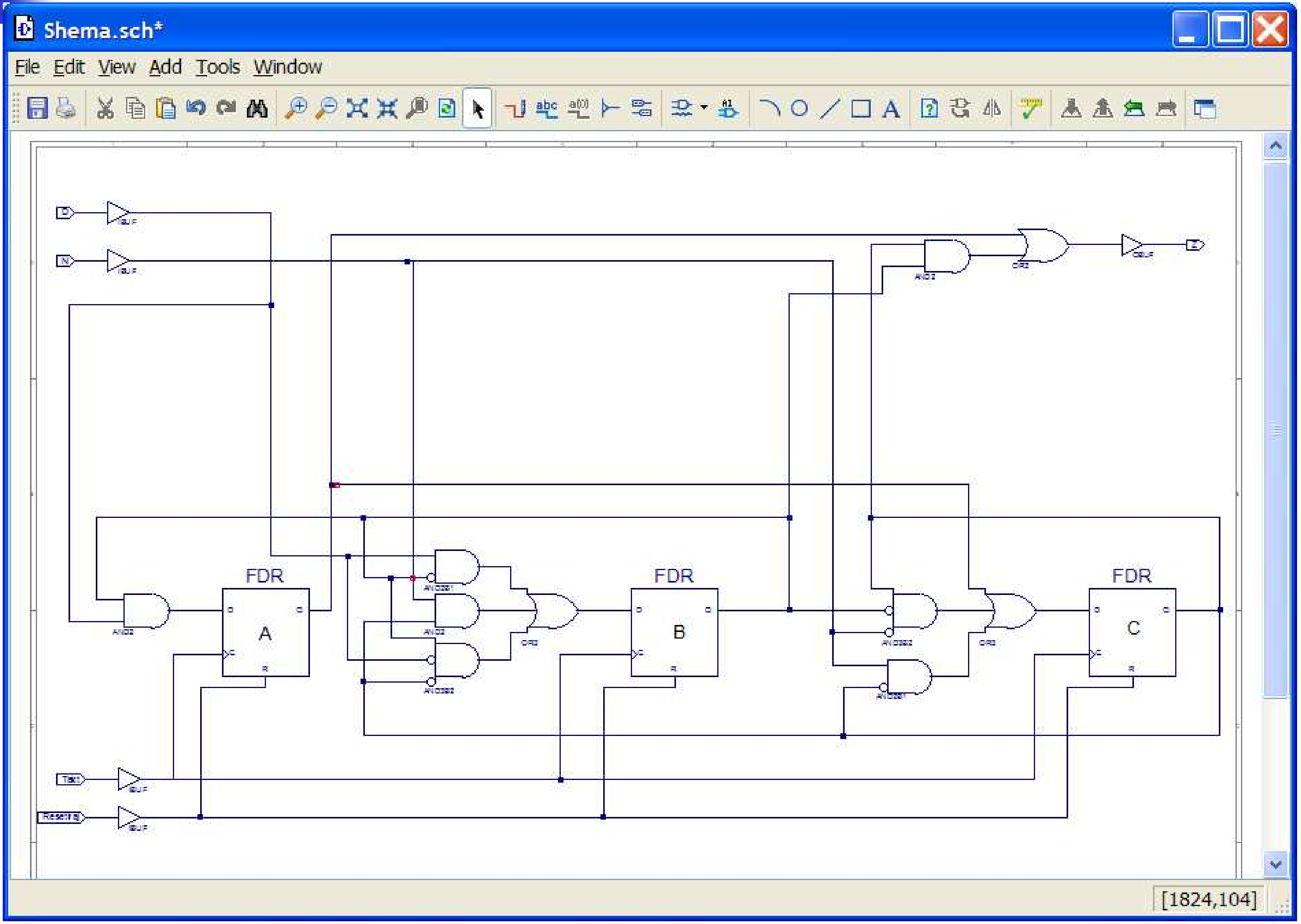
$Z=A+BC$

D	Q^{n+1}
0	0
1	1

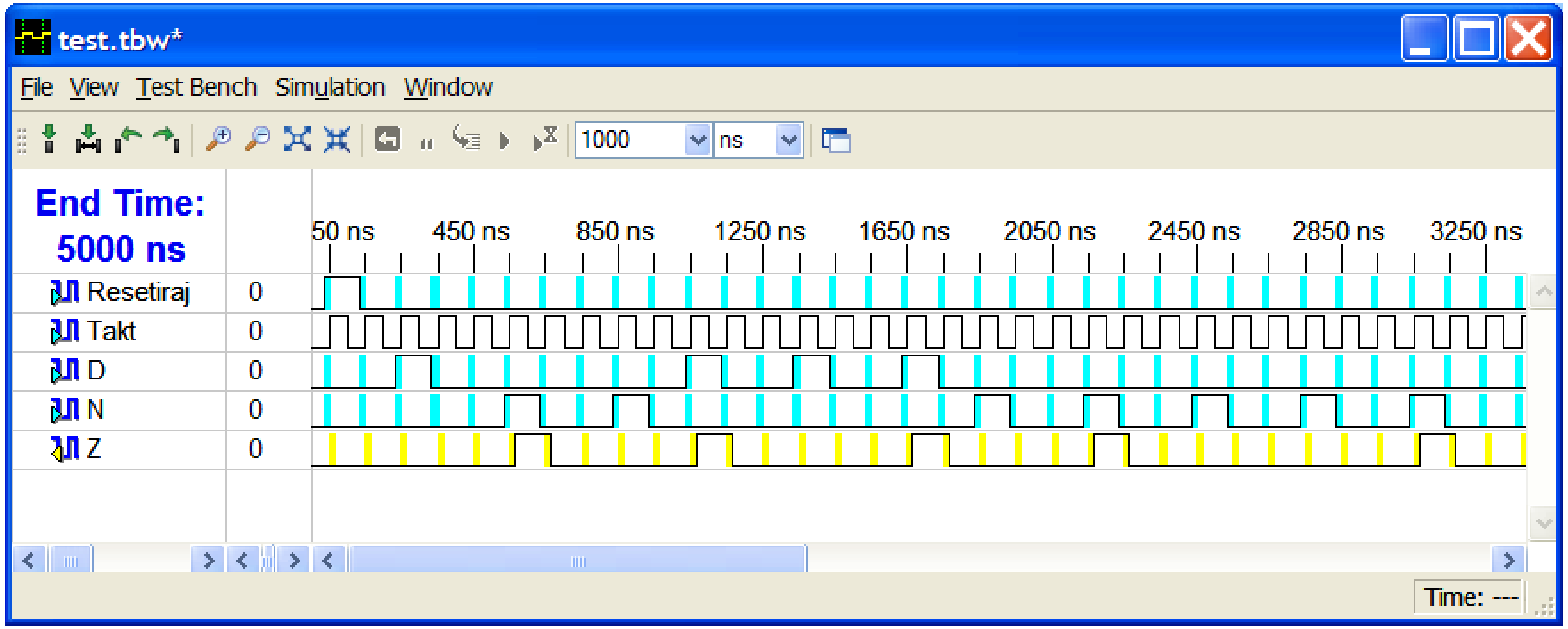
$$Q^{n+1}=D^n$$

Q^n	Q^{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Logička shema



Shema - simulacija



VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Dbist is
   Port ( D : in  STD_LOGIC;
             CP : in  STD_LOGIC;
             Reset : in STD_LOGIC;
             Q : out  STD_LOGIC);
end Dbist;

architecture Behavioral of Dbist is
begin
   process (CP, Reset)
   begin
     if Reset='1' then Q <= '0';
     elsif (CP'event and CP='0') then
         Q <= D;
     end if;
   end process;
end Behavioral;
```

VHDL

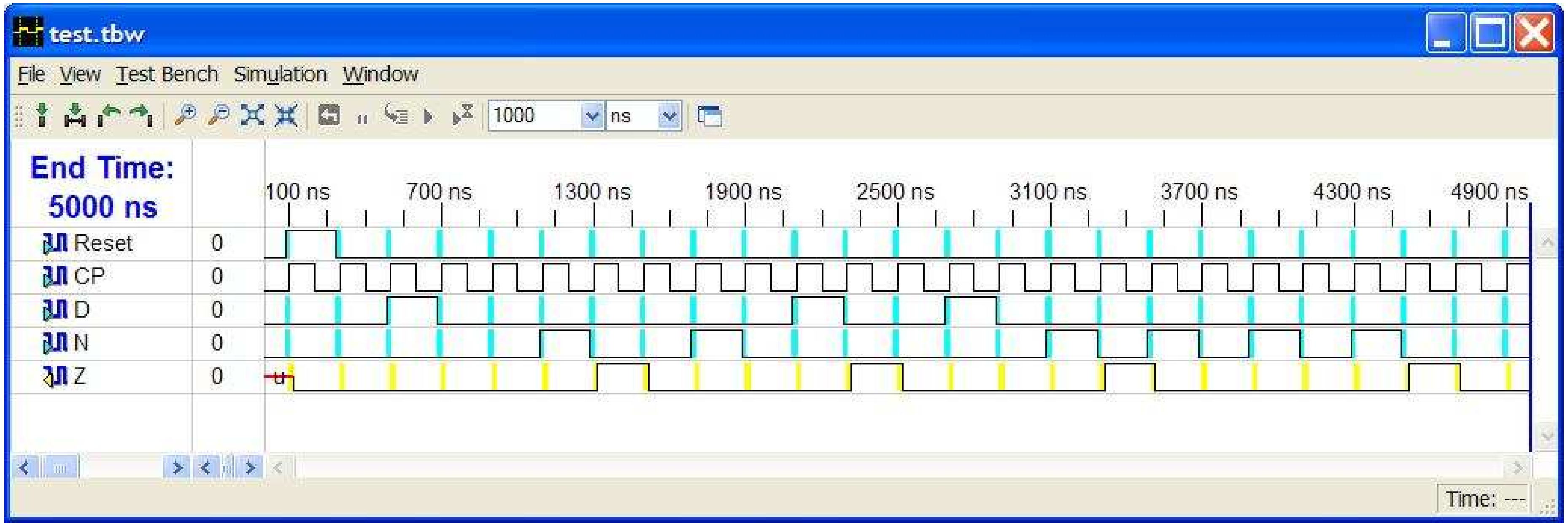
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity VendingMachine is
  Port ( D,N : in  STD_LOGIC;
         CP, Reset : in  STD_LOGIC;
         Z : out  STD_LOGIC);
end VendingMachine;

architecture Struct of VendingMachine is
  signal A,B,C,Anxt,Bnxt,Cnxt : STD_LOGIC;
begin
  Anxt <= B and D;
  Bnxt <= (not B and D) or (C and N) or (B and not C and not D);
  Cnxt <= A or (not B and C and not N) or (not C and N);
  Z <= (not A and B and C) or (A and not B and not C);

  Bist_A : entity Dbist port map (D=>Anxt, Q=>A, Reset=>Reset, CP=>CP);
  Bist_B : entity Dbist port map (D=>Bnxt, Q=>B, Reset=>Reset, CP=>CP);
  Bist_C : entity Dbist port map (D=>Cnxt, Q=>C, Reset=>Reset, CP=>CP);
end Struct;
```

VHDL - simulacija

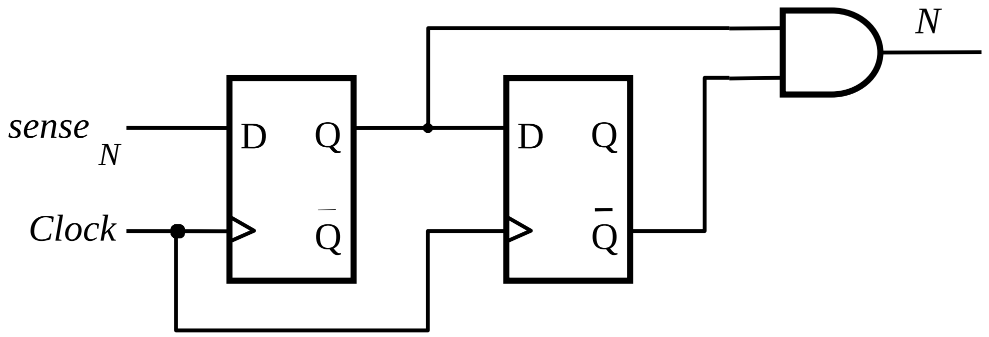


VHDL + D_{sense}, N_{sense}

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity filter is
    Port ( Ulaz : in  STD_LOGIC;
           Takt : in  STD_LOGIC;
           Izlaz : out  STD_LOGIC);
end filter;

architecture struktura of filter is
signal izl1, izl2 : STD_LOGIC;
begin
  Bist1 : entity Dbist port map (D=>Ulaz, Q=>izl1, Reset=>'0', CP=>Takt);
  Bist2 : entity Dbist port map (D=>izl1, Q=>izl2, Reset=>'0', CP=>Takt);
  Izlaz <= izl1 and not izl2;
end struktura ;
```



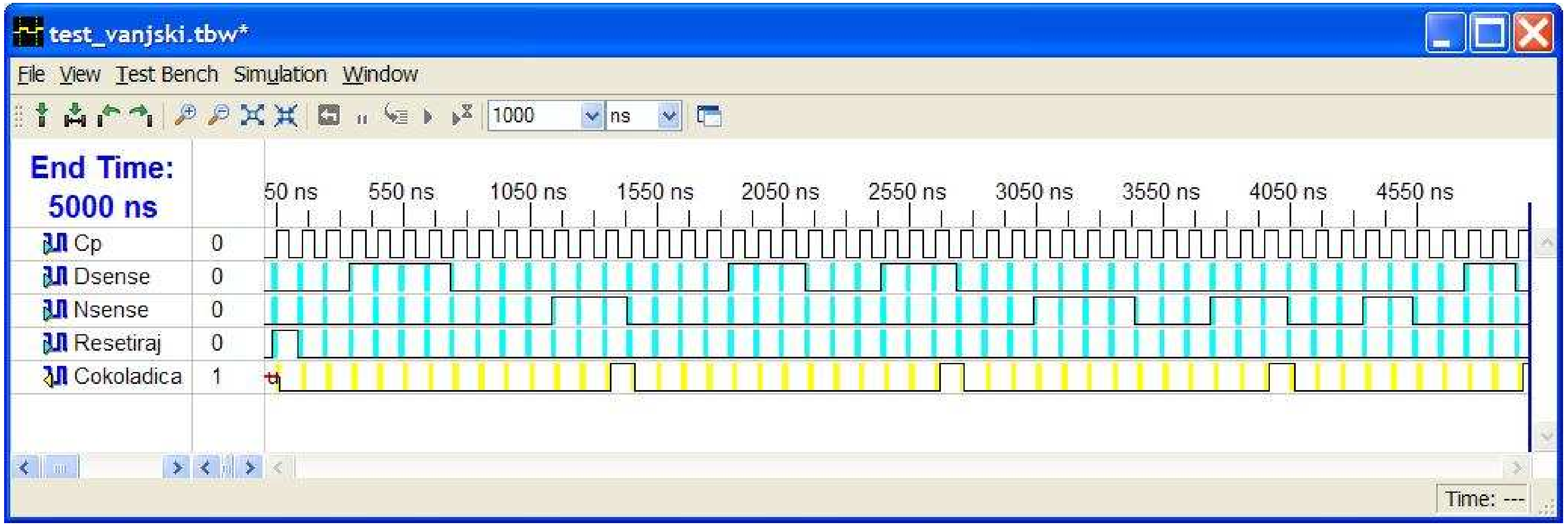
VHDL + D_{sense},N_{sense}

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Automat is
   Port ( Dsense : in  STD_LOGIC;
          Nsense : in  STD_LOGIC;
          Cp : in  STD_LOGIC;
          Resetiraj : in STD_LOGIC;
          Cokoladica : out  STD_LOGIC);
end Automat;

architecture struktura of Automat is
signal D, N : STD_LOGIC;
begin
  FilterD : entity Filter port map (Ulaz=>Dsense, Izlaz=>D, Takt=>Cp);
  FilterN : entity Filter port map (Ulaz=>Nsense, Izlaz=>N, Takt=>Cp);
  VM : entity VendingMachine port map (D=>D, N=>N, CP=>Cp,
                                       Reset=>Resetiraj, Z=>Cokoladica);
end struktura ;
```

VHDL - simulacija



Cp	0
Dsense	0
Nsense	0
Resetiraj	0
Cokoladica	1