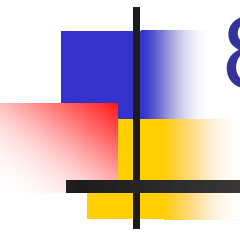


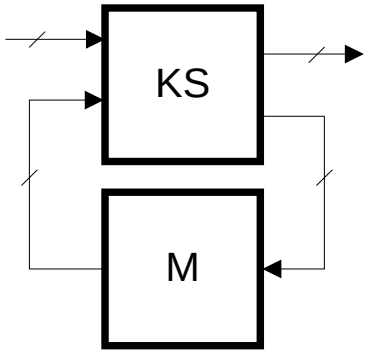
8. Sekvencijski sklopovi

Sadržaj predavanja

- **sinkroni sekvencijski sklopovi**
- kanonski modeli
- projektiranje sekvencijskih sklopova
- izvedbe sekvencijskih sklopova
- analiza sekvencijskih sklopova
- vremenski odnosi

Sinkroni sekvencijski sklopovi

- sekvencijski sklopovi:
  - digitalni sklopovi koji imaju sposobnost pamćenja
  - izlaz je funkcija:
    - trenutnog stanja ulaza
    - trenutnog *unutarnjeg* stanja sklopa
      ~ postoji *memorija*
  - interpretacija memorije (npr. računala)
    ~ pamćenje memorijskih riječi (= višebitni podaci)
    → *registri*



Sinkroni sekvencijski sklopovi

- definicije:
  - *n* bistabila $\rightarrow 2^n$ stanja
    ~ *strojevi stanja* (engl. state machines)
  - stanja je konačno mnogo ($\leq 2^n$)
    ~ strojevi *s konačnim brojem* stanja
    (engl. Finite State Machines, FSM)
  - slijed operacija u sekvencijskom sklopu
    ~ "ugrađeni" algoritam:
    algoritamski stroj stanja
    (engl. Algorithmic State Machine, ASM)
  - operacije se obavljaju bez čovjekove pomoći
    ~ *automat*:
    digitalni automat, *konačni* automat

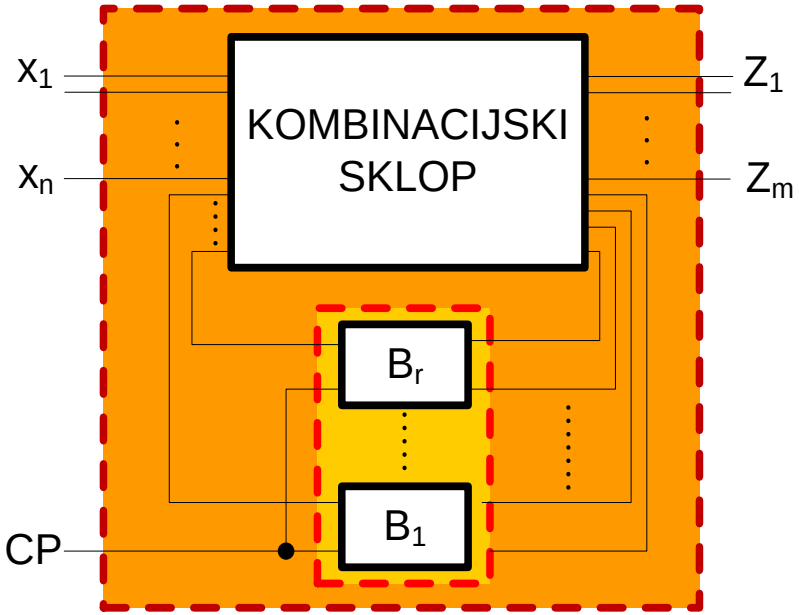
Sinkroni sekvencijski sklopovi

- ograničenje na *sinkrone* sekvencijske sklopove!
  ~ rad (promjena stanja) sinkroniziran s impulsima CP:
  - značajno lakši postupak
    ~ sinkronizacija:
    - na *globalni* CP sustava
    - u odnosu na *najsporiju* stazu/element sustava
      ~ *sporije* od *asinkronih* sekvencijskih sustava
  - vrijeme je *diskretizirano*
    ~ projektiranje svedeno na *kombinacijsko* određivanje:
    - *slijedećeg* stanja sklopa
    - izlaza sklopa

    na temelju *sadašnjeg* stanja sklopa i narinutih ulaza

Sinkroni sekvencijski sklopovi

- općenito strukturiranje sinkronog sekvencijskog sklopa ~ *kanonski* oblik:
  - kombinacijski (pod)sklop
  - memorija s *upravljanim* (= *sinkronim*) bistabilima ~ *registar* = (pod)sklop za pamćenje *višebitnih* podataka

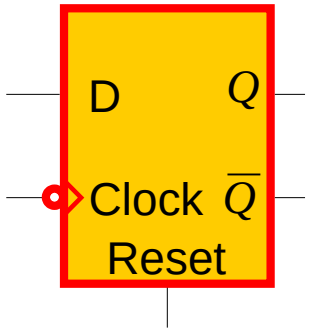


Sinkroni sekvencijski sklopovi

- registar tipično izveden D-bistabilima
  ~ modeliranje u jeziku VHDL

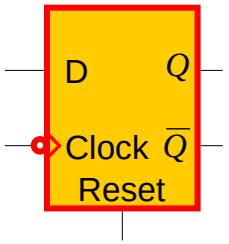
Primjer: VHDL model bridom upravljanog D-bistabila (padajući brid signala takta), sa *sinkronim* ulazom za brisanje

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

ENTITY Dbistabil IS PORT (
  D, Reset: IN std_logic;
  Q, Qn: OUT std_logic;
  Clock: IN std_logic
);
END Dbistabil;
```



Sinkroni sekvencijski sklopovi

```
ARCHITECTURE Behavioral OF Dbistabil IS
BEGIN
  PROCESS(Clock)
    VARIABLE Qint: std_logic;
  BEGIN
    IF falling_edge(Clock) THEN
      IF Reset = '1' THEN
        Qint := '0';
      ELSE
        Qint := D;
      END IF;
    END IF;
    Q <= Qint AFTER 5 ns;
    Qn <= NOT Qint AFTER 5 ns;
  END PROCESS;
END BEHAVIORAL;
```



Sadržaj predavanja

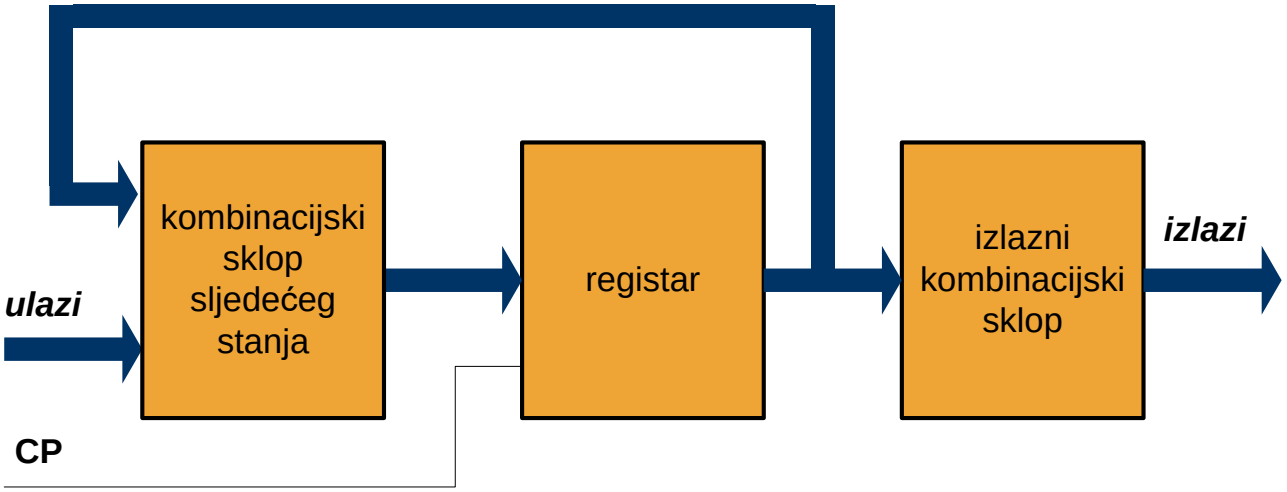
- sinkroni sekvencijski sklopovi
- **kanonski modeli sekvencijskih sklopova**
  - **Mooreov model**
  - **Mealyjev model**
  - **mješoviti model**
- projektiranje sekvencijskih sklopova
- izvedbe sekvencijskih sklopova
- analiza sekvencijskih sklopova
- vremenski odnosi

Kanonski modeli sekvencijskih sklopova

- *više* kanonskih modela opće strukture sinkronog sekvencijskog sklopa; najznačajniji:
  - Mooreov model
  - Mealyjev model
  - mješoviti model

Mooreov model

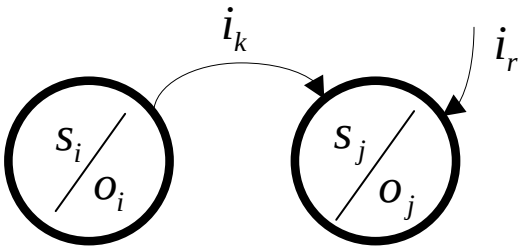
- "automat stanja"
  ~ izlaz ovisi *samo o* unutarnjem stanju



$A = \langle I, O, S, \delta, \mu \rangle$

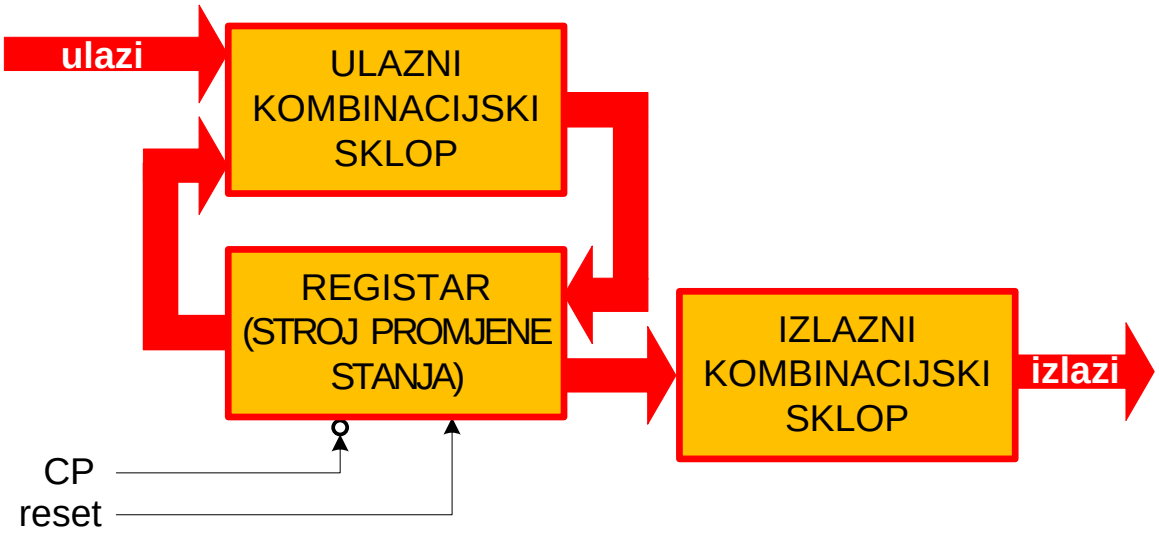
$\delta: S \times I \rightarrow S$

$\mu: S \rightarrow O$



Mooreov model

- modeliranje osnovne strukture Mooreovog stroja s konačnim brojem stanja u jeziku VHDL:



Mooreov model

- deklaracija tipova i modeliranje sučelja:

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

ENTITY automatMoore IS PORT (
  ulazi:   IN std_logic_vector(N DOWNTO 0);
  reset:   IN std_logic; -- inicijalizira početno stanje
  izlazi:  OUT std_logic_vector(M DOWNTO 0);
  clock:   IN std_logic
);
END automatMoore;
```

Mooreov model

- definicija arhitekture:
  - kodiranje stanja *intuitivno*
    ~ prirodnim binarnim kodom (binarni brojevi)

```
ARCHITECTURE ponasanje OF automatMoore IS
  SIGNAL state_present, state_next: std_logic_vector(K DOWNTO 0);
  CONSTANT S0: std_logic_vector(K DOWNTO 0) := "0...000";
  CONSTANT S1: std_logic_vector(K DOWNTO 0) := "0...001";
           S2: ...
  -- kodna riječ pojedinog stanja definirana lokalnom konstantom
BEGIN
  -- ulazni kombinacijski sklop računa sljedeće stanja
  -- iz ulaza i trenutnog stanja
  PROCESS(ulazi, state_present)
  BEGIN
    -- sljedeće stanje sklopa određeno signalima iz liste
    -- osjetljivosti; npr. bezuvjetni prijelaz u stanje S0:
    state_next <= S0;
  END PROCESS;
```

1

Mooreov model

```
-- izlazni kombinacijski sklop računa izlaz sklopa
-- na temelju trenutnog stanja
PROCESS(state_present)
BEGIN
  -- izlaz sklopa određen iz trenutnog stanja:
  CASE state_present IS
    WHEN S0 => izlazi <= ...;
    WHEN S1 => izlazi <= ...;
    -- itd. ...
    WHEN OTHERS => izlaz <= ...;
  END CASE;
END PROCESS;
```

2

Mooreov model

```
  -- stroj promjene stanja mijenja stanje na temelju signala
  -- clock i asinkronih ulaza
  PROCESS(clock, reset)
  BEGIN
    -- asinkroni ulaz(i) dominira(ju)
    IF reset = '1' THEN
      state_present <= S0;
    -- inače djelovanje sinkronih ulaza
    ELSIF falling_edge(clock) THEN
      state_present <= state_next;
    END IF;
  END PROCESS;

END ponasanje;
```

3

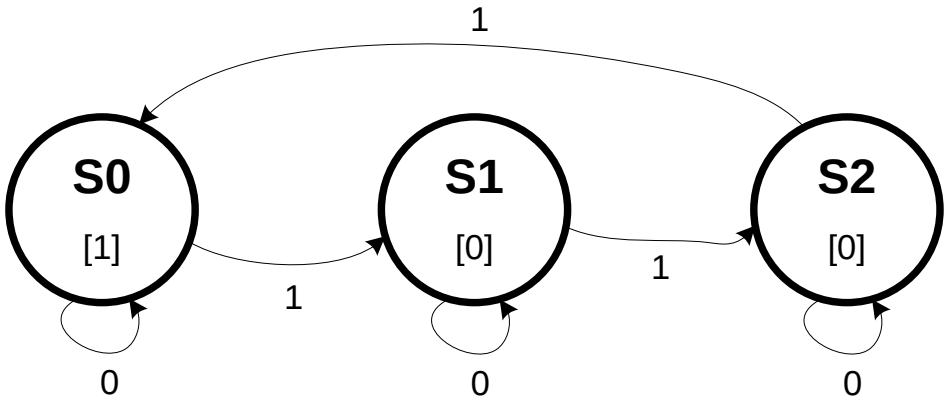
Mooreov model

- alternativni (bolji?) način definiranja stanja:
  → navesti *oznake* stanja, ali ne i način kodiranja

```
TYPE stateType IS (S0, S1, S2, S3);
SIGNAL state_present, state_next: stateType;
```

Mooreov model

Primjer: VHDL model Mooreovog stroja s konačnim brojem stanja




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

ENTITY automat1 IS PORT (
  input0: IN std_logic;
       y: OUT std_logic;
   clock: IN std_logic);
END automat1;
```

Mooreov model

```
ARCHITECTURE Behavioral OF automat1 IS
  SIGNAL state_present, state_next: std_logic_vector(1 DOWNTO 0);
  CONSTANT S0: std_logic_vector(1 DOWNTO 0) := "11";
  CONSTANT S1: std_logic_vector(1 DOWNTO 0) := "01";
  CONSTANT S2: std_logic_vector(1 DOWNTO 0) := "10";
BEGIN
  PROCESS(input0, state_present)
  BEGIN
    CASE state_present IS
      WHEN S0 =>   IF input0 = '0' THEN state_next <= S0;
                                   ELSE state_next <= S1;
                   END IF;
      WHEN S1 =>   IF input0 = '0' THEN state_next <= S1;
                                   ELSE state_next <= S2;
                   END IF;
      WHEN S2 =>   IF input0 = '0' THEN state_next <= S2;
                                   ELSE state_next <= S0;
                   END IF;
      WHEN OTHERS =>                    state_next <= S0;
    END CASE;
  END PROCESS;
```

1

Mooreov model

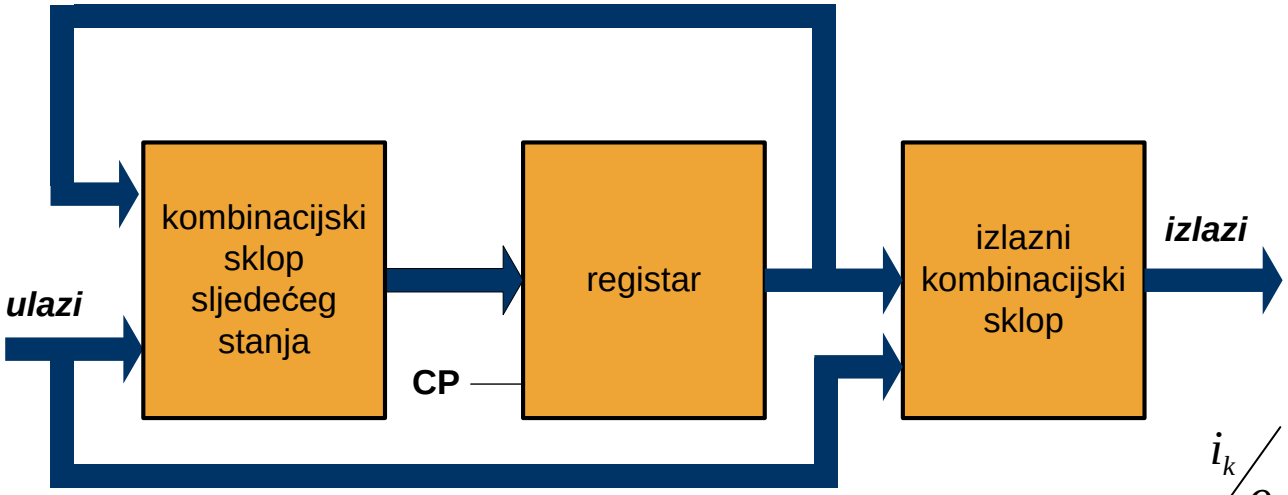
```
  PROCESS(state_present)
  BEGIN
    CASE state_present IS
      WHEN S0 => y <= '1';
      WHEN S1 => y <= '0';
      WHEN S2 => y <= '0';
      WHEN OTHERS => y <= '0';
    END CASE;
  END PROCESS;

  PROCESS(clock)
  BEGIN
    IF falling_edge(clock) THEN
       state_present <= state_next;
    END IF;
  END PROCESS;

END BEHAVIORAL;
```

2

3

Mealyjev model

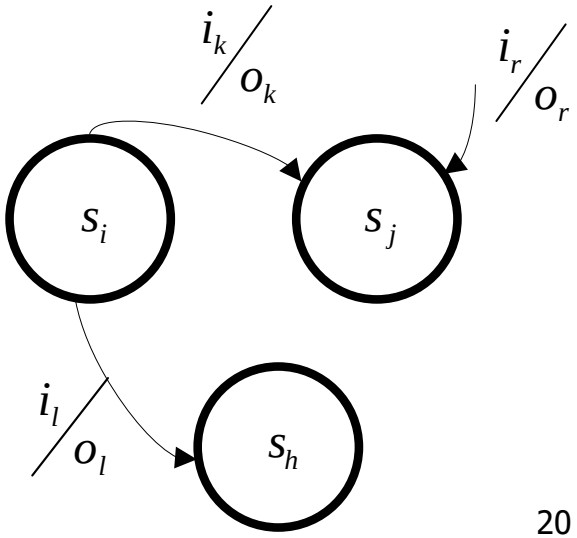
- *Mealyjev model* : "automat prijelaza"
  ~ izlaz ovisi o unutarnjem stanju, *ali i o ulazu*



$A = \langle I, O, S, \delta, \lambda \rangle$

$\delta: S \times I \rightarrow S$

$\lambda: S \times I \rightarrow O$



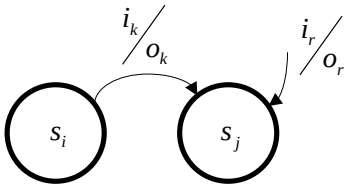
Mealyjev model

- ekvivalencija Mealyjevog i Mooreovog automata:

$A = \langle I, O, S, \delta, \lambda \rangle$

$\delta: S \times I \rightarrow S$

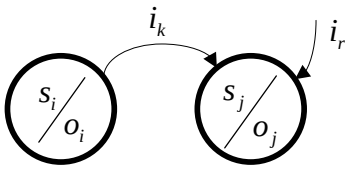
$\lambda: S \times I \rightarrow O$



$A = \langle I, O, S, \delta, \mu \rangle$

$\delta: S \times I \rightarrow S$

$\mu: S \rightarrow O$



$$q^{n+1} = \delta(q^n, x^n)$$

$$z^n = \lambda(q^n, x^n)$$

$$q^{n+1} = \delta(q^n, x^n)$$

$$z^n = \mu(q^n)$$

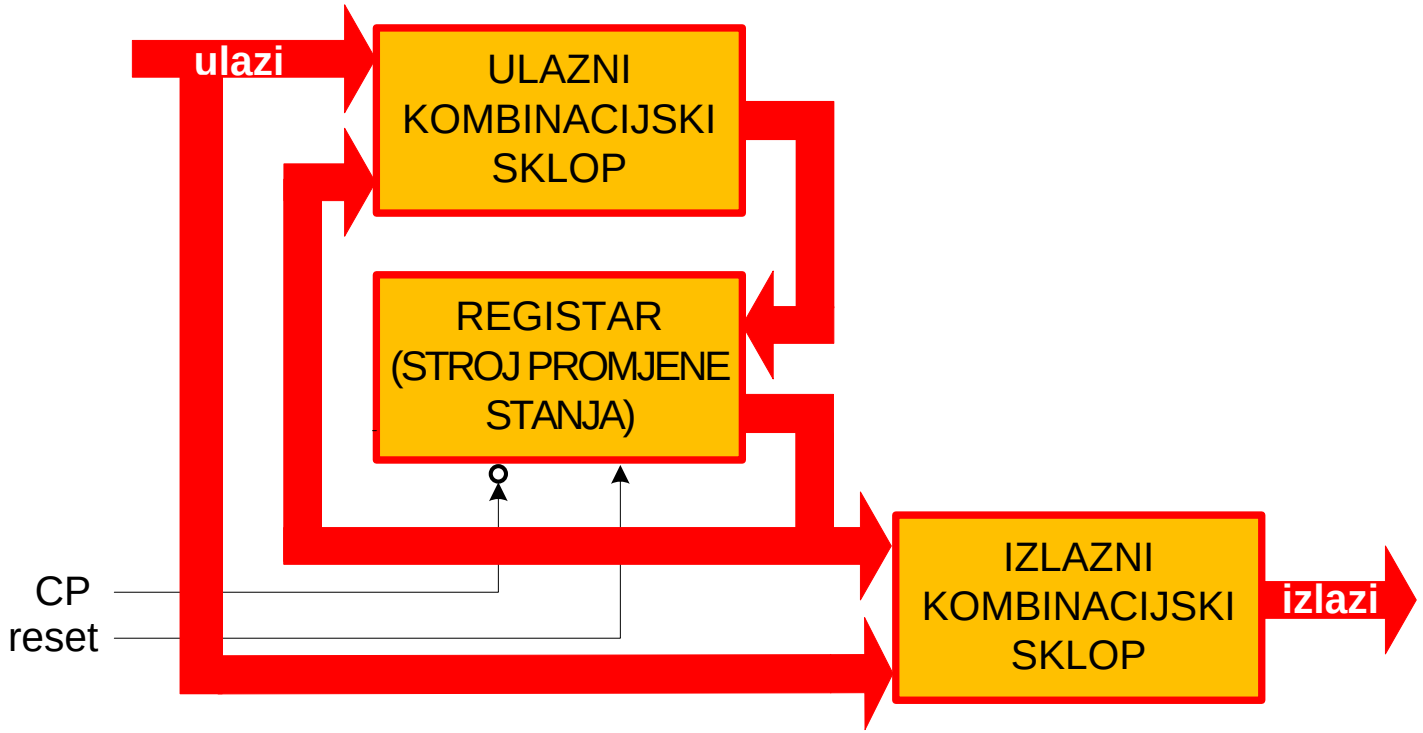
$$= \mu(\delta(q^{n-1}, x^{n-1}))$$

$$= \mu'(q^{n-1}, x^{n-1})$$

- izlaz Mooreovog automata ovisi o *prethodnom* unutarnjem stanju i ulazu!
- moguć prijelaz iz jednog u drugi

Mealyjev model

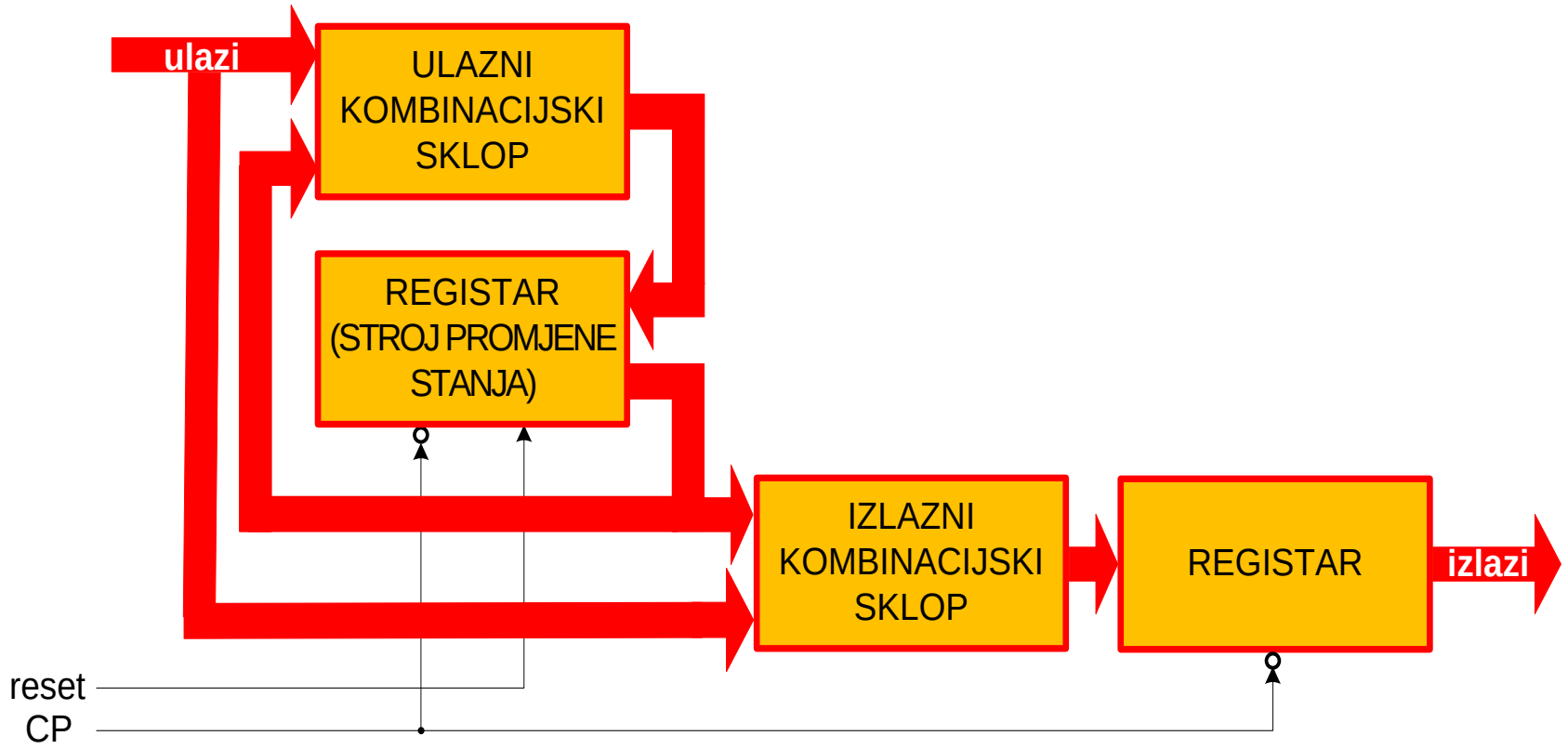
- modeliranje osnovne strukture Mealyjevog stroja s konačnim brojem stanja u jeziku VHDL:



- osigurati da promjena ulaza ne utječe na promjenu izlaza prije promjene stanja (nailaska CP)
  ~ *sinkronizacija* promjene!

Mealyjev model

- sinkronizacija promjene izlaza
  ~ osiguranje stabilne vrijednosti izlaza između dva susjedna impulsa CP: stanje izlaza pamti se *dodatnim* registrom



Mealyjev model

- deklaracija tipova i modeliranje sučelja:

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

ENTITY automatMealy IS PORT (
  ulazi: IN std_logic_vector(N DOWNTO 0);
  reset: IN std_logic;
  izlazi: OUT std_logic_vector(M DOWNTO 0);
  clock: IN std_logic
);
END automatMealy;
```

Mealyjev model

- definicija arhitekture:

```
ARCHITECTURE ponasanje OF automatMealy IS
  SIGNAL state_present, state_next: std_logic_vector(K DOWNTO 0);
  CONSTANT S0: std_logic_vector(K DOWNTO 0) := "0...000";
  CONSTANT S1: std_logic_vector(K DOWNTO 0) := "0...001";
  ...
  SIGNAL izlazi_next: std_logic_vector(M DOWNTO 0);
BEGIN
  -- ulazni kombinacijski sklop računa sljedeće stanje i izlaz
  -- iz ulaza i trenutnog stanja
  PROCESS(ulazi, state_present)
  BEGIN
    -- sljedeće stanje sklopa i izlaz određeni iz liste
    -- osjetljivosti; npr. bezuvjetni prijelaz u stanje S0
    -- i sve '0' na izlazima:
    state_next <= S0;
    izlazi_next <= "000...000";  -- niz od (M+1) nule.
  END PROCESS;
```

1

2

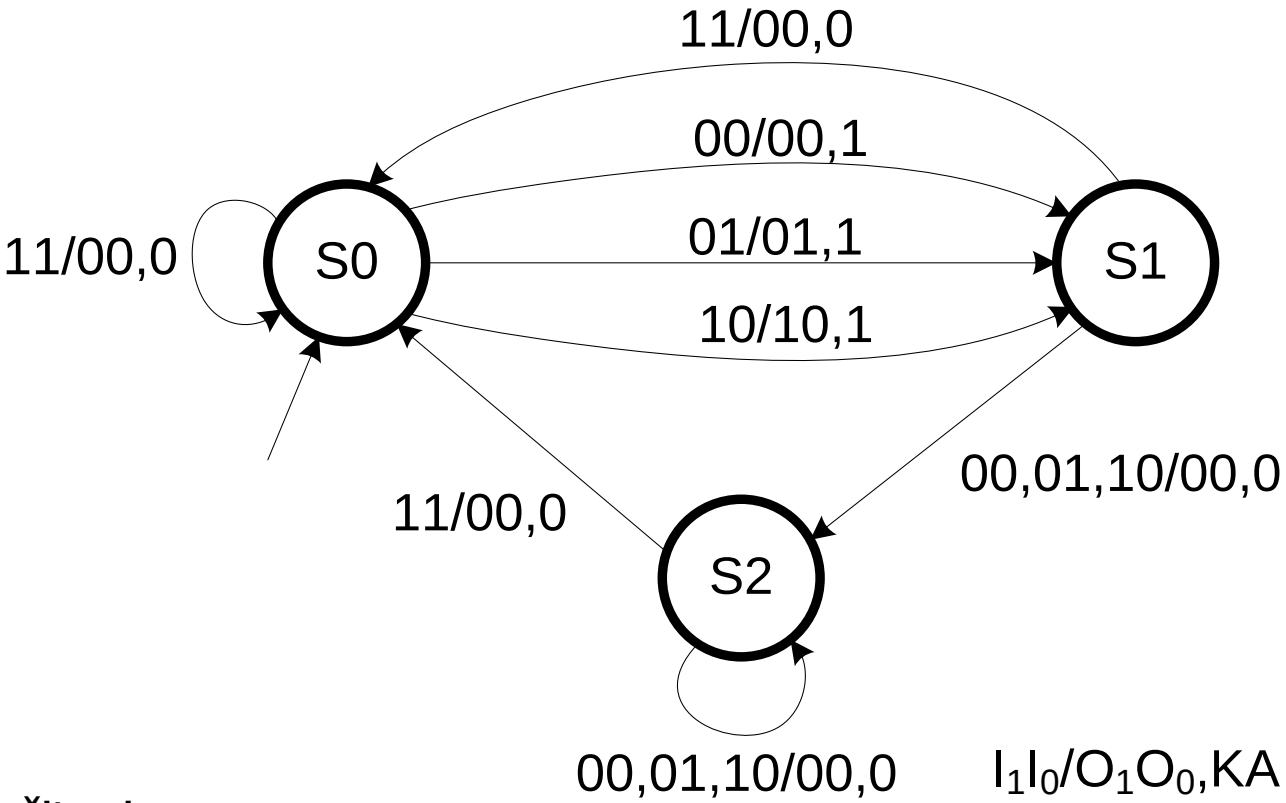
Mealyjev model

```
  -- stroj promjene stanja i registar; promjena stanja i izlaza
  -- na temelju signala clock i asinkronih ulaza
PROCESS(clock, reset)
  BEGIN
    -- asinkroni ulaz(i) dominira(ju)
    IF reset = '1' THEN
      state_present <= S0;  -- postaviti stanje u S0
      izlaz <= "000...00";  -- postaviti izlaze u npr. '0'
    -- inače djelovanje sinkronih ulaza
                  ELSIF falling_edge(clock) THEN
                        state_present <= state_next;
                        izlazi <= izlazi_next;
    END IF;
  END PROCESS;

END ponasanje;
```

3

Mealyjev model

Primjer: VHDL model Mealyjevog stroja
s konačnim brojem stanja



legenda:

I_1I_0: ulazi

O_1O_0: izlazi

KA: izlaz očitanja

Mealyjev model

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

ENTITY automat2 IS PORT (
  i1,i0 : IN std_logic;
  o1 : OUT std_logic; o0 : OUT std_logic;
  ka : OUT std_logic;
  clock: IN std_logic);
END automat2;

ARCHITECTURE Behavioral OF automat2 IS
  SIGNAL state_present, state_next: std_logic_vector(1 DOWNTO 0);
  SIGNAL o1_next, o0_next, ka_next: std_logic;
  CONSTANT S0: std_logic_vector(1 DOWNTO 0) := "00";
  CONSTANT S1: std_logic_vector(1 DOWNTO 0) := "01";
  CONSTANT S2: std_logic_vector(1 DOWNTO 0) := "10";
BEGIN
```

Mealyjev model

```
PROCESS(i1, i0, state_present)
   VARIABLE pom: std_logic_vector(1 DOWNTO 0);
 BEGIN
   pom := (i1, i0);
   CASE state_present IS
     WHEN S0 =>
         CASE pom IS
           WHEN "00" =>   state_next <= S1; o1_next <= '0';
                          o0_next <= '0'; ka_next <= '1';
           WHEN "01" =>   state_next <= S1; o1_next <= '0';
                          o0_next <= '1'; ka_next <= '1';
           WHEN "10" =>   state_next <= S1; o1_next <= '1';
                          o0_next <= '0'; ka_next <= '1';
           WHEN "11" =>   state_next <= S0; o1_next <= '0';
                          o0_next <= '0'; ka_next <= '0';
           WHEN OTHERS => state_next <= S0; o1_next <= '0';
                          o0_next <= '0'; ka_next <= '0';
         END CASE;
```

1

2

Mealyjev model

```
WHEN S1 =>
  CASE pom IS
    WHEN "00" =>    state_next <= S2; o1_next <= '0';
                    o0_next <= '0'; ka_next <= '0';
    WHEN "01" =>    state_next <= S2; o1_next <= '0';
                    o0_next <= '0'; ka_next <= '0';
    WHEN "10" =>    state_next <= S2; o1_next <= '0';
                    o0_next <= '0'; ka_next <= '0';
    WHEN "11" =>    state_next <= S0; o1_next <= '0';
                    o0_next <= '0'; ka_next <= '0';
    WHEN OTHERS => state_next <= S0; o1_next <= '0';
                    o0_next <= '0'; ka_next <= '0';
       END CASE;
```

Mealyjev model

```
        WHEN S2 =>
          CASE pom IS
             WHEN "00" =>    state_next <= S2; o1_next <= '0';
                             o0_next <= '0'; ka_next <= '0';
             WHEN "01" =>    state_next <= S2; o1_next <= '0';
                             o0_next <= '0'; ka_next <= '0';
             WHEN "10" =>    state_next <= S2; o1_next <= '0';
                             o0_next <= '0'; ka_next <= '0';
             WHEN "11" =>    state_next <= S0; o1_next <= '0';
                             o0_next <= '0'; ka_next <= '0';
             WHEN OTHERS => state_next <= S0; o1_next <= '0';
                             o0_next <= '0'; ka_next <= '0';
              END CASE;
        WHEN OTHERS =>
            state_next <= S0; o1_next <= '0';
            o0_next <= '0'; ka_next <= '0';
 END CASE;
END PROCESS;
```

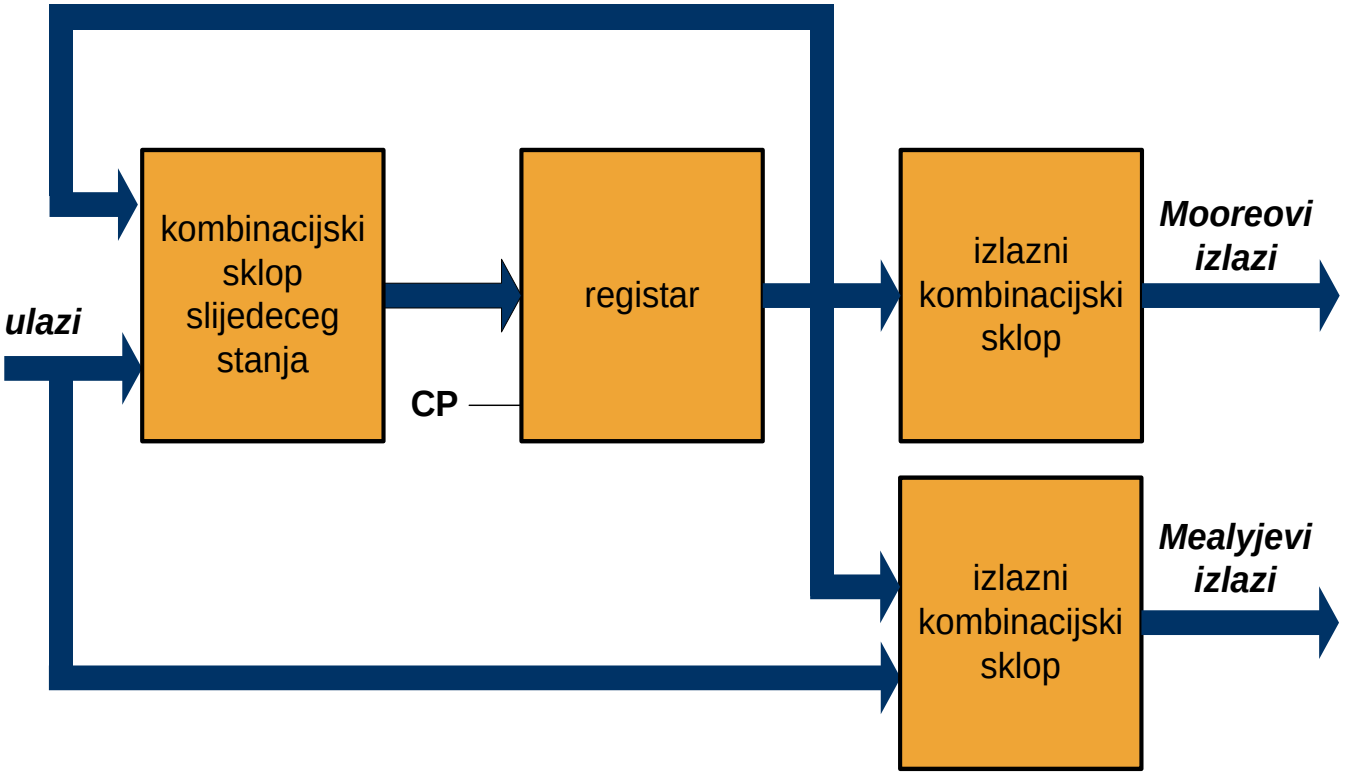
Mealyjev model

```
 PROCESS(clock)
  BEGIN
    IF falling_edge(clock) THEN
       state_present <= state_next;
      o1 <= o1_next;
      o0 <= o0_next;
      ka <= ka_next;
    END IF;
  END PROCESS;

END Behavioral;
```

3

Mješoviti model

- *mješoviti model*
  ~ izdvojeni izlazi Mooreovog i Mealyjevog modela



Sadržaj predavanja

- sinkroni sekvencijski sklopovi
- kanonski modeli sekvencijskih sklopova
- **projektiranje sekvencijskih sklopova**
  - **minimiziranje memorije**
  - **kodiranje stanja**
  - **ostvarivanje sekvencijskog sklopa**
- izvedbe sekvencijskih sklopova
- analiza sekvencijskih sklopova
- vremenski odnosi

Projektiranje sekvencijskih sklopova

- neformalni opis postupka (1):
  1. specifikacija sekvencijskog sklopa
     ~ ulazni jezik: verbalno, algoritamski, ...
  2. izrada dijagrama stanja ili tablice stanja:
     ~ m: broj stanja, n: broj bistabila $\quad 2^{n-1} < m \leq 2^n$
  3. minimiziranje broja stanja $\rightarrow$ smanjenje broja bistabila
     ~ minimizacija memorije!
  4. kodiranje stanja
     ~ dodjela binarne kodne riječi pojedinom stanju:
     - prikladno pridruživanje
       ~ minimiziranje kombinacijskog (pod)sklopa
     - težak kombinatorni problem

Projektiranje sekvencijskih sklopova

- neformalni opis postupka (2):

5. izbor tipa bistabila:
   - dobivanje *ulaznih* jednadžbi bistabila
     ~ pobuda potrebna za odgovarajući prijelaz
     ($\rightarrow$ generiranje slijedećeg stanja)
   - dobivanje *izlaznih* jednadžbi sklopa
   - minimizacija kombinacijskog (pod)sklopa
6. formalni zapis sekvencijskog sklopa
   npr. *logička shema*
   ~ izbor tehnologije ostvarenja (SIC, ASIC, ...)

Minimiziranje memorije

- standardni pristup
  ~ *Huffman-Mealyjeva metoda*:
  - ovdje: za *potpuno specificirane* sklopove
    ~ *za svako* je unutarnje stanje definirano slijedeće unutarnje stanje + izlaz
  - minimizacija broja bistabila
    ~ redukcija broja unutarnjih stanja nalaženjem *ekvivalentnih* (nerazlučivih) stanja
  - *ekvivalentna* stanja:
    ~ ona iz kojih se *istom* pobudom (ulazni niz simbola) dobiva *isti* izlaz (izlazni niz simbola)

Minimiziranje memorije

- ideja Huffman-Mealyjeve metode:
  - klasa ekvivalentnih stanja zamjenjuje stanja koja su ekvivalentna
    ~ automat s *reduciranim* brojem unutarnjih stanja
    $\Rightarrow$ *minimizirana* memorija
  - početni (ne-minimalni) i konačni (minimalni) automat
    ~ *ekvivalentni* s obzirom na *izvana opazivo ponašanje* (engl. externally observable behavior):
    jednaki odziv na jednaku pobudu

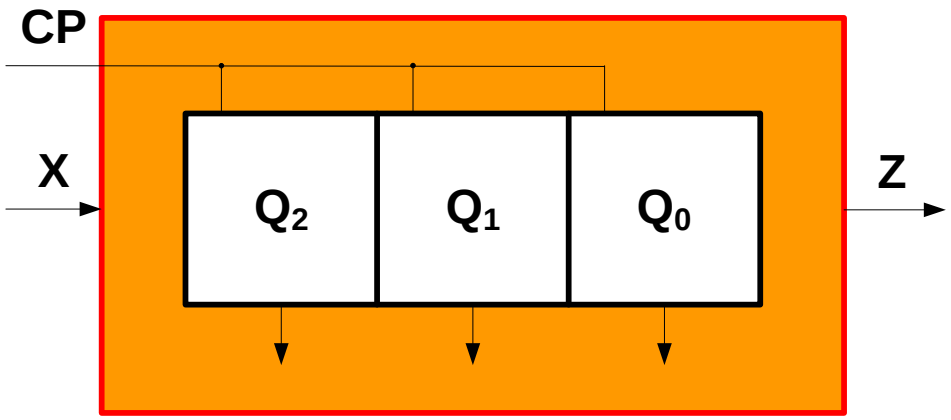
Minimiziranje memorije

- algoritam Huffman-Mealyjeve metode:
  - podjela unutarnjih stanja u *najmanji mogući broj* klasa ekvivalentnih stanja,
    tako da stanja u istoj klasi imaju *iste izlaze*
    ~ stanja grupirati s obzirom na izlaze
  - *daljnja podjela* dobivenih klasa na podklase,
    tako da prijelazi iz stanja jedne te iste klase vode u stanja jedne druge iste klase
  - u konačnici se prijelazi između stanja zamjenjuju prijelazima između *klasa* stanja
    ~ naravno, klasa ima *manje* od stanja ☺

Minimiziranje memorije

Primjer: Huffman-Mealyjeva metoda

- sekvencijski sklop s jednim ulazom x, jednim izlazom z i 8 stanja
- $8 = 2^3$ stanja $\rightarrow$ 3 bistabila Q_2, Q_1, Q_0
  ~ promatraju se Q_i (promjena stanja) + izlaz
- rad sekvencijskog sklopa
  ~ tablica stanja

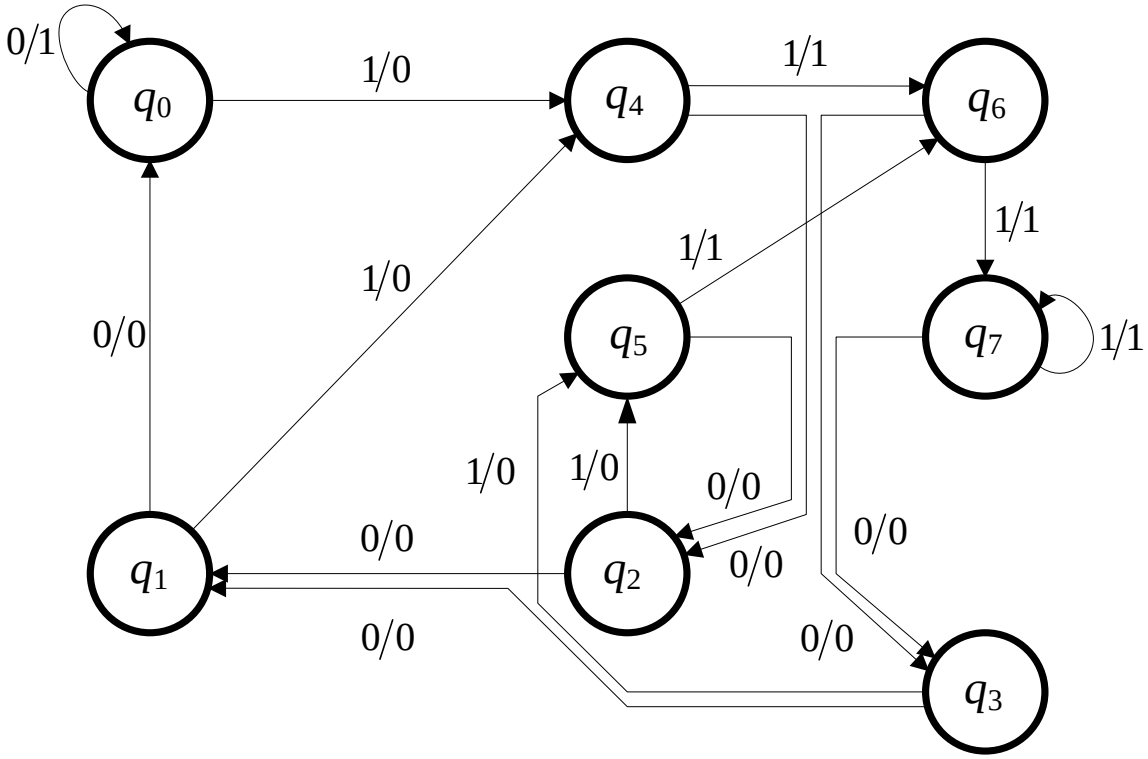


q^n	q^{n+1}, z^n	
	$x^n=0$	$x^n=1$
q_0	q_0,1	q_4,0
q_1	q_0,0	q_4,0
q_2	q_1,0	q_5,0
q_3	q_1,0	q_5,0
q_4	q_2,0	q_6,1
q_5	q_2,0	q_6,1
q_6	q_3,0	q_7,1
q_7	q_3,0	q_7,1

Minimiziranje memorije

- specifikacija automata:

q^n	q^{n+1}, z^n	
	$x^n=0$	$x^n=1$
q_0	$q_0,1$	$q_4,0$
q_1	$q_0,0$	$q_4,0$
q_2	$q_1,0$	$q_5,0$
q_3	$q_1,0$	$q_5,0$
q_4	$q_2,0$	$q_6,1$
q_5	$q_2,0$	$q_6,1$
q_6	$q_3,0$	$q_7,1$
q_7	$q_3,0$	$q_7,1$



Minimiziranje memorije

- klase ekvivalentnih stanja prema stanju izlaza
  ~ (početno) 3 klase

  $a = \{q_0\}$
  $b = \{q_1, q_2, q_3\}$
  $c = \{q_4, q_5, q_6, q_7\}$

q^n	q^{n+1}, z^n	
	$x^n=0$	$x^n=1$
q_0	$q_0,1$	$q_4,0$
q_1	$q_0,0$	$q_4,0$
q_2	$q_1,0$	$q_5,0$
q_3	$q_1,0$	$q_5,0$
q_4	$q_2,0$	$q_6,1$
q_5	$q_2,0$	$q_6,1$
q_6	$q_3,0$	$q_7,1$
q_7	$q_3,0$	$q_7,1$

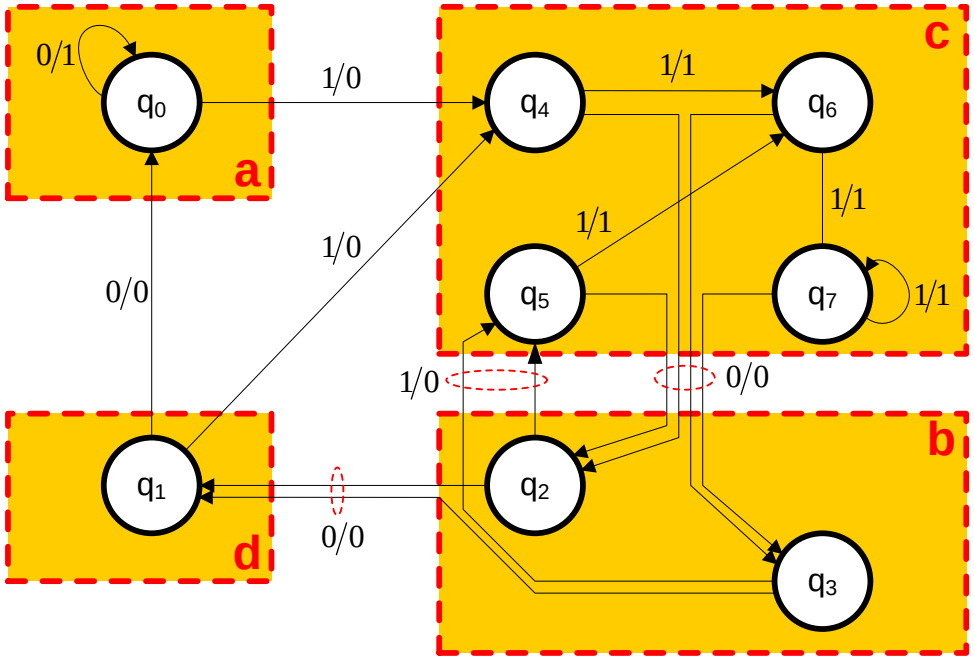
- provjera prijelaza
  ~ 4 klase ekvivalentnih stanja

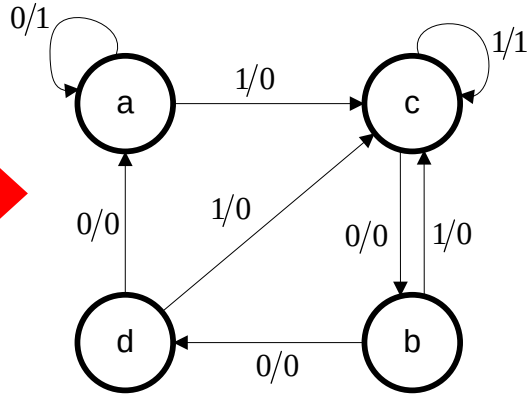
klasa	a	b			c			
stanje	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
sl. klasa	a c	a c	b c	b c	b c	b c	b c	b c

klasa	a	b		c				d
stanje	q_0	q_2	q_3	q_4	q_5	q_6	q_7	q_1
sl. klasa	a c	d c	d c	b c	b c	b c	b c	a c

Minimiziranje memorije

- dijagram stanja:





- nova tablica stanja

stanja u klasi ekvivalencije	q^n	q^{n+1},z^n	
		$x^n=0$	$x^n=1$
q_0	a	a,1	c,0
q_2,q_3	b	d,0	c,0
q_4,q_5,q_6,q_7	c	b,0	c,1
q_1	d	a,0	c,0

Kodiranje stanja

- kodiranje stanja
  ~ pridruživanje binarne kodne riječi pojedinom stanju:
  - utječe na veličinu kombinacijskog sklopa
  - težak kombinatorni problem
    ~ prihvatljiva podoptimalna rješenja
  - trivijalno kodiranje
    ~ prirodni binarni kod

Kodiranje stanja

Primjer: kodiranje stanja

- *kombinirana* tablica stanja
  ~ prijelaz + izlaz u ovisnosti o pobudi (ulazu)
- 4 stanja $\rightarrow$ 2 bistabila (B_1, B_0)
- trivijalno kodiranje
  ~ binarni kod

q^n	q^{n+1}		z^n	
	$x^n=0$	$x^n=1$	$x^n=0$	$x^n=1$
a	a	c	1	0
b	d	c	0	0
c	b	c	0	1
d	a	c	0	0

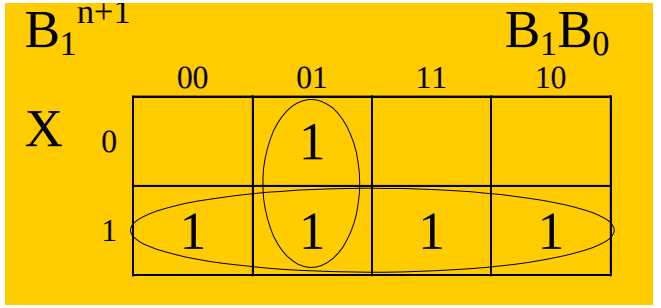
$a \sim 00$
$b \sim 01$
$c \sim 10$
$d \sim 11$

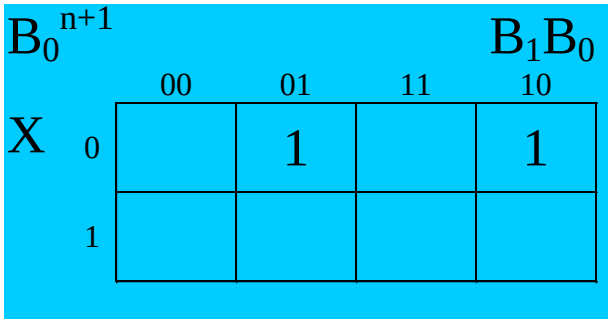
$(B_1B_0)^n$	$(B_1B_0)^{n+1}$		z^n	
	x=0	x=1	x=0	x=1
00	00	10	1	0
01	11	10	0	0
10	01	10	0	1
11	00	10	0	0

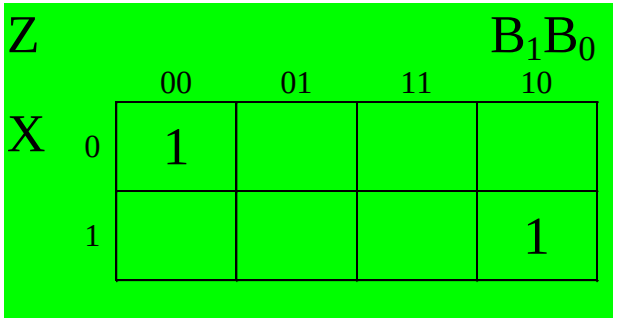
Ostvarivanje sekvencijskog sklopa

Primjer: implementacija memorije D-bistabilima

$(B_1B_0)^n$	$(B_1B_0)^{n+1}$				z^n	
	x=0		x=1		x=0	x=1
00	0	0	1	0	1	0
01	1	1	1	0	0	0
10	0	1	1	0	0	1
11	0	0	1	0	0	0







Ostvarivanje sekvencijskog sklopa

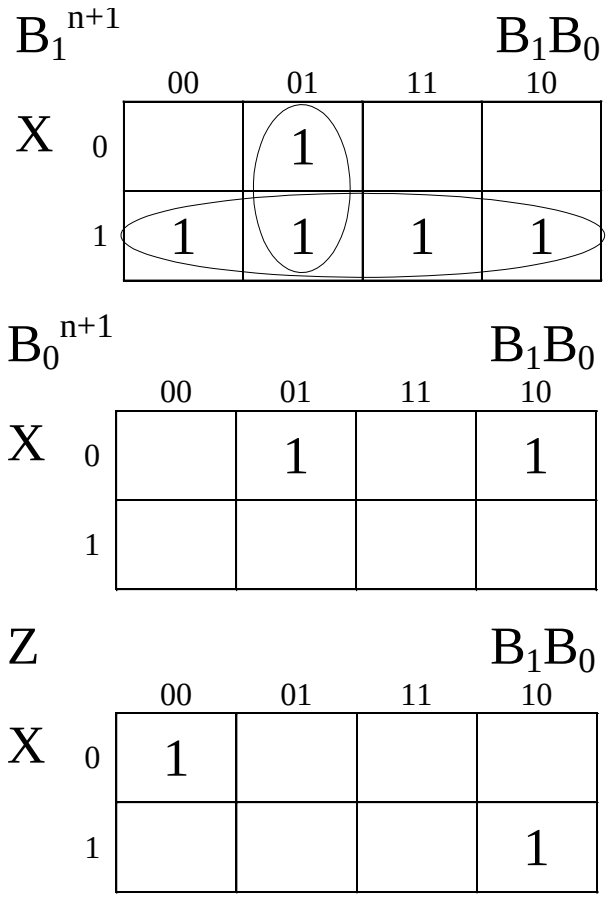
- D bistabili
  ~ posebno jednostavno dobivanje *ulazne jednadžbe* iz *karakteristične* :

$$B^{n+1} = D^n \Rightarrow D^n = B^{n+1}$$

$$D_1 = X + \overline{B}_1 B_0$$

$$D_0 = \overline{X} \cdot \left(\overline{B}_1 B_0 + B_1 \overline{B}_0\right) = \overline{X} \cdot \left(B_1 \oplus B_0\right)$$

$$Z = \overline{B}_1 \overline{B}_0 \overline{X} + B_1 \overline{B}_0 X = \overline{B}_0 \cdot \left(\overline{B}_1 \overline{X} + B_1 X\right) = \overline{B}_0 \cdot \left(B_1 \otimes X\right) = \overline{B}_0 \cdot \overline{B_1 \oplus X}$$



Ostvarivanje sekvencijskog sklopa

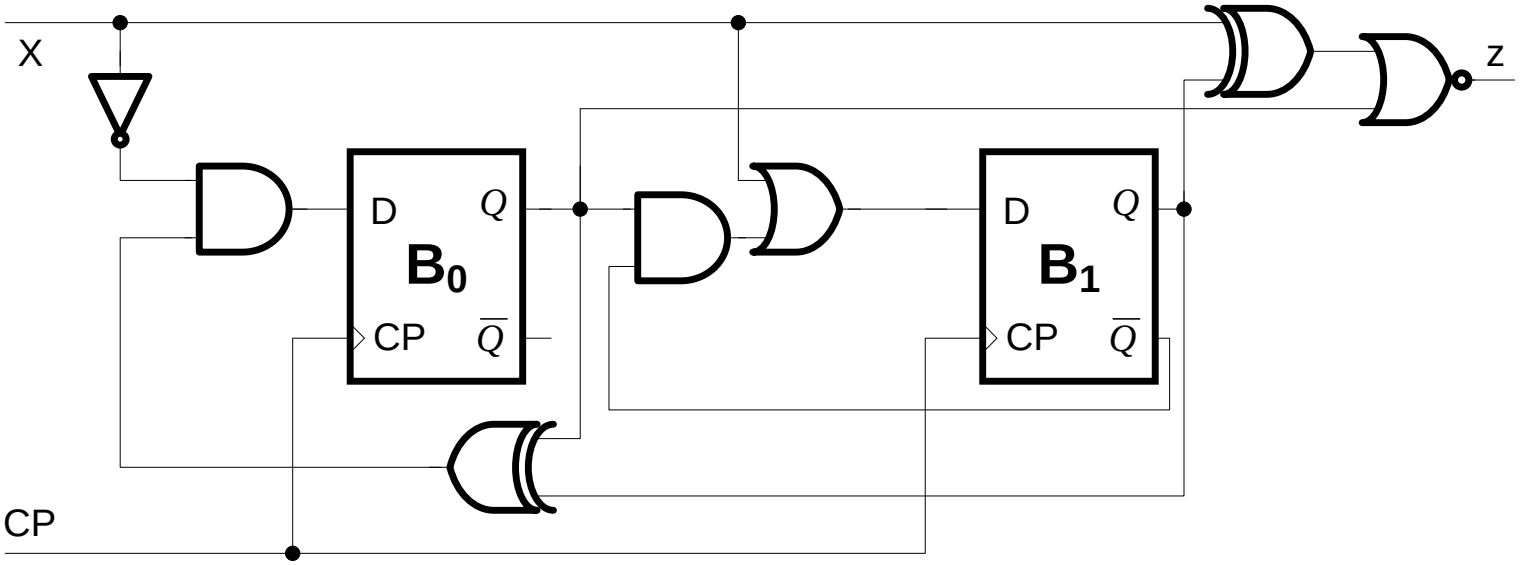
- logička shema:

$$D_1 = X + \overline{B}_1 B_0$$

$$D_0 = \overline{X} \cdot \left(\overline{B}_1 B_0 + B_1 \overline{B}_0 \right)$$

$$= \overline{X} \cdot \left(B_1 \oplus B_0 \right)$$

$$Z = \overline{B}_1 \overline{B}_0 \overline{X} + B_1 \overline{B}_0 X = \overline{B_0 + (B_1 \oplus X)}$$



Ostvarivanje sekvencijskog sklopa

Primjer: izvedba memorije JK-bistabilima

- koristi se *uzbudna tablica*:

		Q^n		Q^{n+1}	J	K
α	:=	0	→	1	1	x
β	:=	1	→	0	x	1
0	:=	0	→	0	0	x
1	:=	1	→	1	x	0

oznaka vrste prijelaza

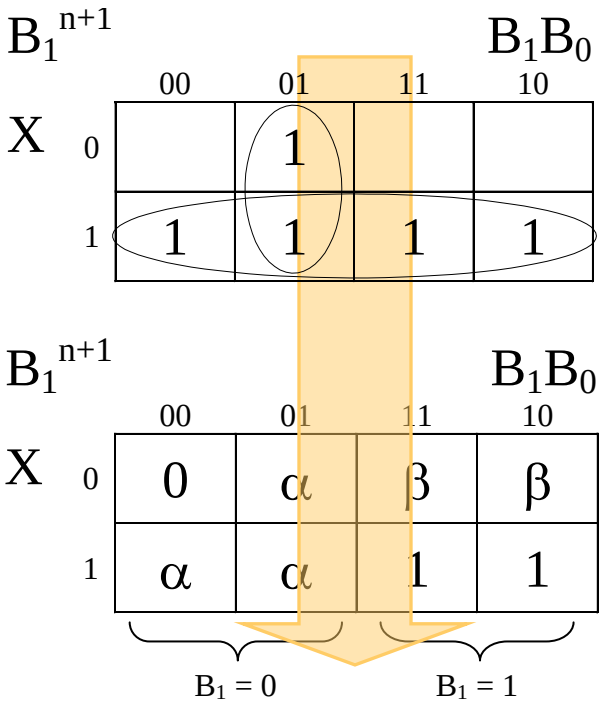
vrsta prijelaza

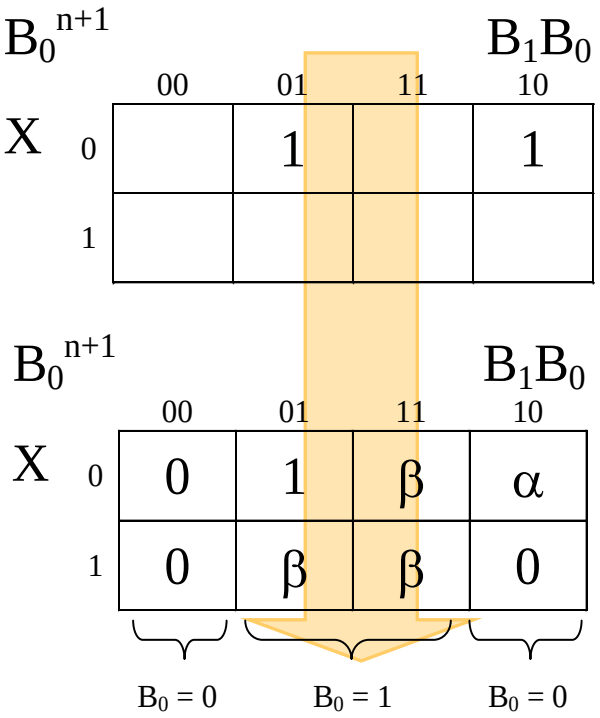
potrebna uzbuda JK-bistabila

Ostvarivanje sekvencijskog sklopa

- označavanje prijelaza zapisom *uzbudne tablice*:

Q^n	Q^{n+1}		J	K
0	1	α	1	x
1	0	β	x	1
0	0	0	0	x
1	1	1	x	0

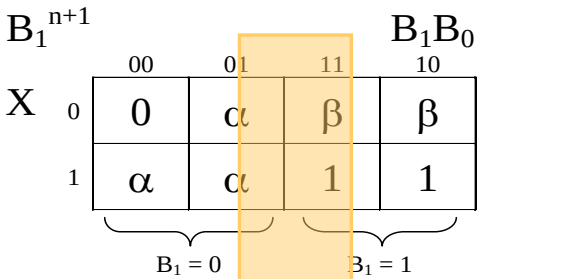


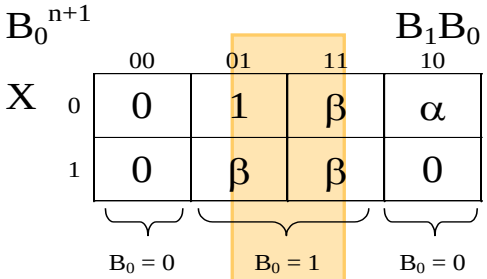


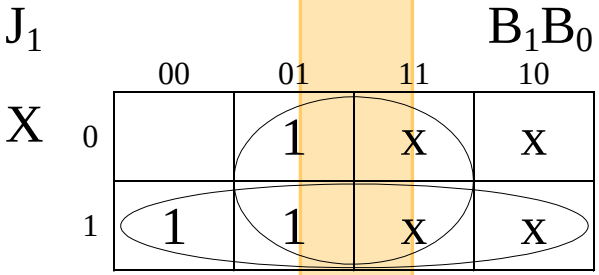
Ostvarivanje sekvencijskog sklopa

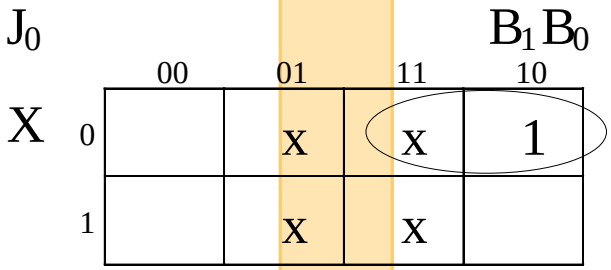
- izvođenje ulaznih jednadžbi:

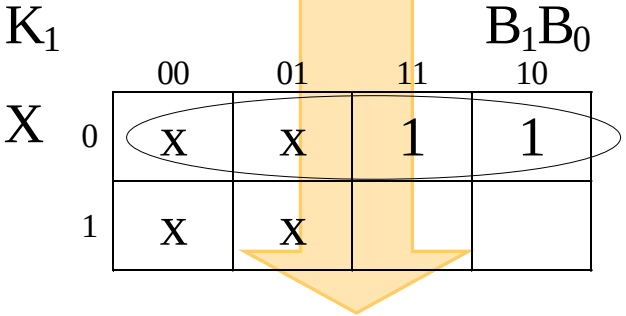
Q^n		Q^{n+1}		J	K
0	→	1	: α	1	x
1	→	0	: β	x	1
0	→	0	: 0	0	x
1	→	1	: 1	x	0

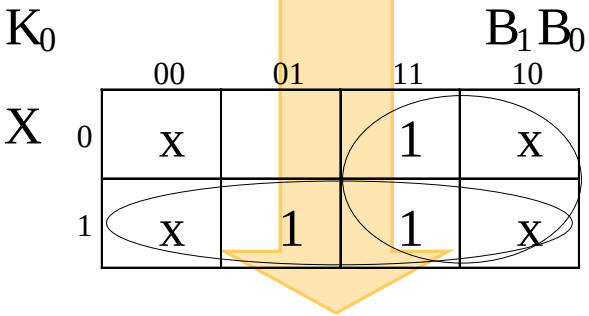






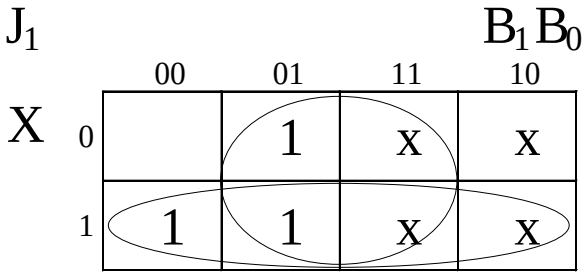


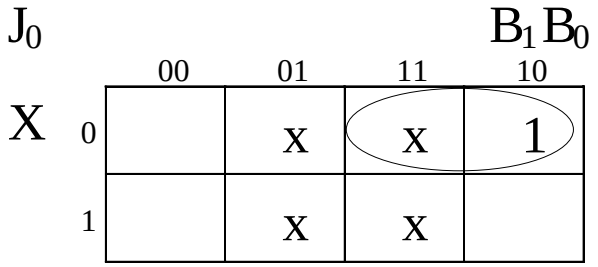


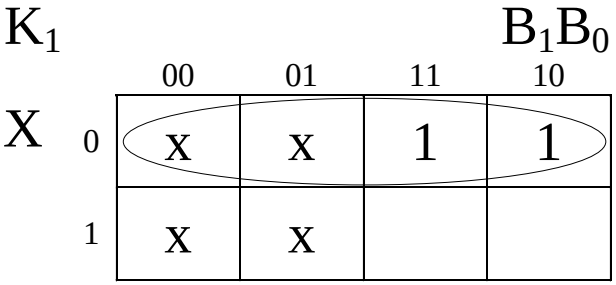


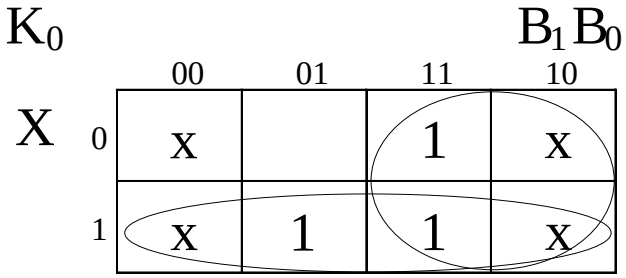
Ostvarivanje sekvencijskog sklopa

- ulazne jednadžbe:









$$J_1 = B_0 + X$$
$$K_1 = \overline{X}$$

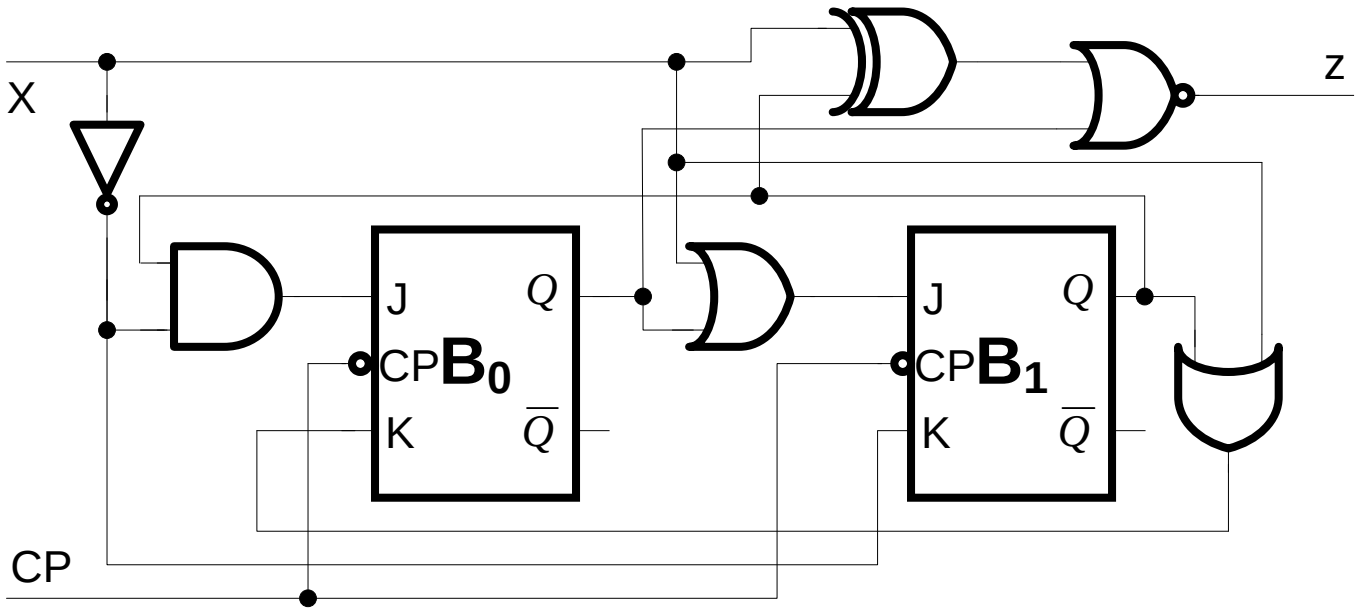
$$J_0 = B_1 \overline{X}$$
$$K_0 = B_1 + X$$

Ostvarivanje sekvencijskog sklopa

- logička shema:

$$J_0 = B_1\overline{X} \qquad J_1 = B_0 + X \qquad Z = \overline{B_0 + (B_1 \oplus X)}$$

$$K_0 = B_1 + X \qquad K_1 = \overline{X}$$



Sadržaj predavanja

- sinkroni sekvencijski sklopovi
- kanonski modeli sekvencijskih sklopova
- projektiranje sekvencijskih sklopova
- **izvedbe sekvencijskih sklopova**
- analiza sekvencijskih sklopova
- vremenski odnosi

Izvedbe sekvencijskih sklopova

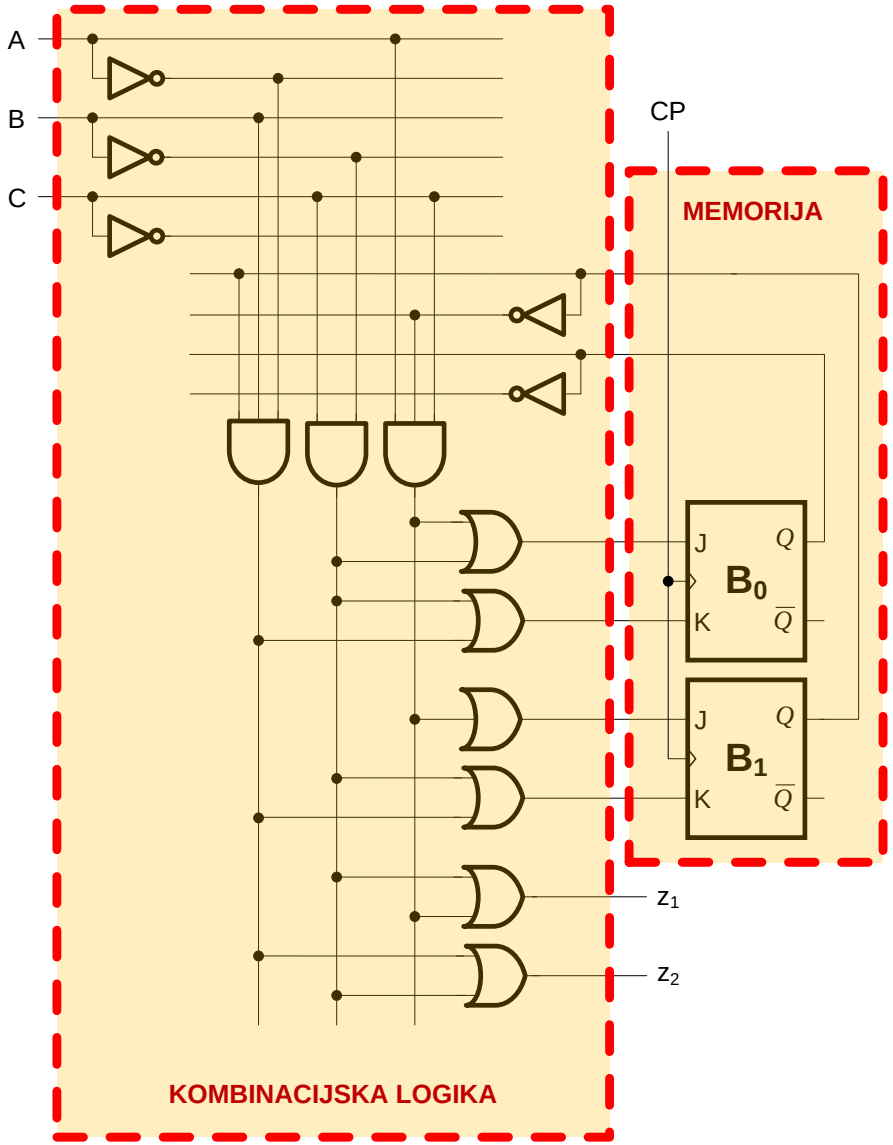
- mogućnosti izvedbe
  ~ raspoloživa "tehnologija":
  - bistabili (D, RS, JK)
    + kombinacijska logika (osnovni logički sklopovi, univerzalni sklopovi)
    ~ kao prije ☺
  - sekvencijski moduli
    ~ moduli koje sadrže kombinacijski sklop *i* memoriju (niz/skup bistabila ili registara)

Izvedbe sekvencijskih sklopova

- *sekvencijski moduli* :
  ~ općenita klasifikacija:
  - *standardni* moduli:
    ~ *n*–bitni moduli za ostvarivanje određene karakteristične funkcije (*fiksno* "ožičeni"):
    - brojila: upravljačka funkcija (brojanje)
    - registri: funkcija pohranjivanja podataka
  - *univerzalni* moduli
    ~ ostvarivanje *proizvoljnih* sekvencijskih sklopova (*programirljivi* su!):
    memorija + *programirljivi* (kombinacijski) sklop;
    generiranje kombinacijske logike slično generiranju Booleovih funkcija kombinacijskim modulima

Izvedbe sekvencijskih sklopova

- primjer univerzalnog sekvencijskog modula
  ~ *programirljivo sekvencijsko polje*
  (engl. sequential PLA)



Sadržaj predavanja

- sinkroni sekvencijski sklopovi
- kanonski modeli sekvencijskih sklopova
- projektiranje sekvencijskih sklopova
- izvedbe sekvencijskih sklopova
- **analiza sekvencijskih sklopova**
- vremenski odnosi

Analiza sekvencijskih sklopova

- analiza *sinkronog* sekvencijskog sklopa
  ~ *obrnuti* postupak:
  - *ponašanje* (= rad) postojećeg sklopa?
  - *formalni* opis
- formalizmi poznati od prije:
  - tablica stanja
    ~ prijelazi u sljedeća stanja, izlazi
  - dijagram stanja
    ~ grafički prikaz tablice stanja
  - jednadžbe stanja
    - iz tablice stanja ili direktno iz logičke sheme
    - opis uvjeta za promjenu stanja bistabila:

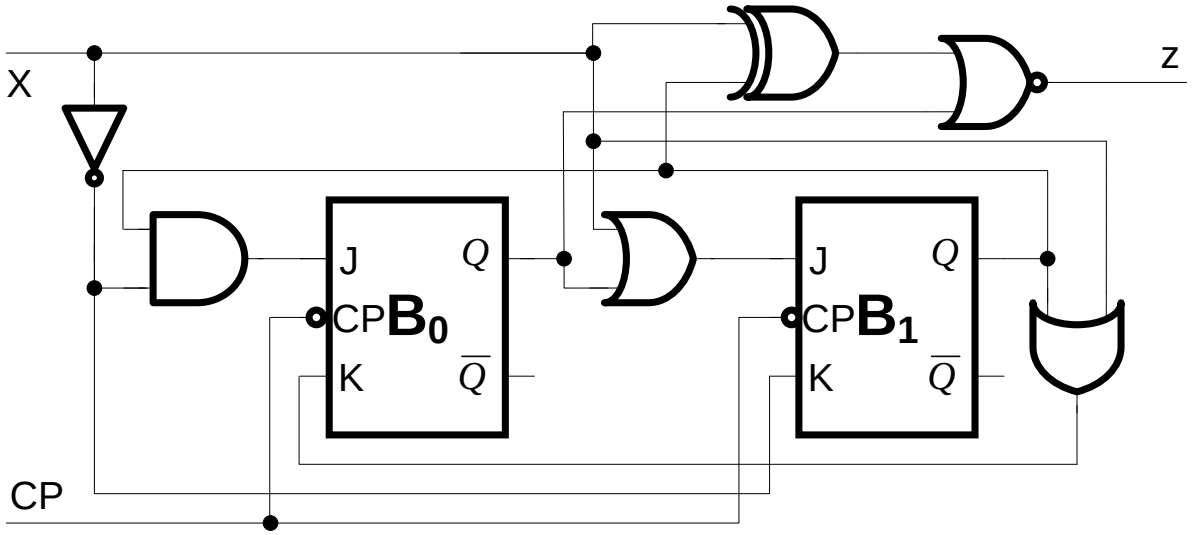
$$B_i^{n+1} = f\left(B_0^n, B_1^n, \ldots, B_{r-1}^n, x_0, \ldots, x_{l-1}\right)$$

Analiza sekvencijskih sklopova

- uobičajeni postupak analize:
  - očitati *logičku shemu* iz samog sklopa
  - iz *logičke sheme* izvesti:
    - *ulazne* jednadžbe za svaki bistabil
      ~ Booleov izraz koji utvrđuje potrebnu pobudu za željeno ponašanje bistabila
    - *izlazne* jednadžbe za svaki izlaz
  - iz ulaznih jednadžbi bistabila i izlaznih jednadžbi ispisati *tablicu stanja*
    ~ početno stanje = neko karakteristično stanje
    npr. kodna riječ $0_{10} = 000...0_2$

Analiza sekvencijskih sklopova

Primjer: analiza prethodnog sklopa



iz sheme sklopa očitano:

$$J_1 = B_0 + X \qquad J_0 = B_1\overline{X}$$

$$K_1 = \overline{X} \qquad K_0 = X + B_1$$

$$Z = \overline{B_0 + (B_1 \oplus X)} = \overline{B}_1\overline{B}_0\overline{X} + B_1\overline{B}_0X$$

Analiza sekvencijskih sklopova

$$J_1 = B_0 + X \qquad J_0 = B_1\overline{X} \qquad Z = \overline{B}_1\overline{B}_0\overline{X} + B_1\overline{B}_0X$$

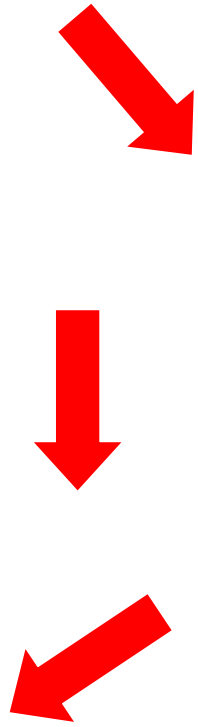
$$K_1 = \overline{X} \qquad K_0 = X + B_1$$

- pridruživanje:

B_1	B_0	stanje
0	0	a
0	1	b
1	0	c
1	1	d

- tablica stanja:

	X=0	X=1
a	a,1	c,0
b	d,0	c,0
c	b,0	c,1
d	a,0	c,0



n							n+1		
B_1	B_0	X	J_1	K_1	J_0	K_0	B_1	B_0	Z
0	0	0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	1	0	0
0	1	0	1	1	0	0	1	1	0
0	1	1	1	0	0	1	1	0	0
1	0	0	0	1	1	1	0	1	0
1	0	1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0	0	0
1	1	1	1	0	0	1	1	0	0

Sadržaj predavanja

- sinkroni sekvencijski sklopovi
- kanonski modeli sekvencijskih sklopova
- projektiranje sekvencijskih sklopova
- izvedbe sekvencijskih sklopova
- analiza sinkronih sekvencijskih sklopova
- **vremenski odnosi**
  - **dinamički parametri bistabila**
  - **dinamički parametri sekvencijskog sklopa**

Vremenski odnosi

- karakteristični dinamički parametri bistabila:
  - maksimalna frekvencija rada
  - vrijeme kašnjenja
  - vrijeme postavljanja
  - vrijeme otpuštanja
  - vrijeme pridržavanja
- karakteristični dinamički parametri sekvencijskog sklopa:
  - maksimalna frekvencija rada
  - raskorak

Vremenski odnosi

- *maksimalna frekvencija* rada bistabila, f_{max}:
  ~ *najveća* frekvencija CP,
  a da bistabil *sigurno* mijenja stanje
  kad to ulazi zahtijevaju
- *vrijeme kašnjenja* bistabila, t_d:
  ~ interval od djelotvorne promjene na ulazu do promjene na izlazu:
  - asinkroni ulazi: u odnosu na S_d, C_d
  - sinkroni ulazi: u odnosu na CP

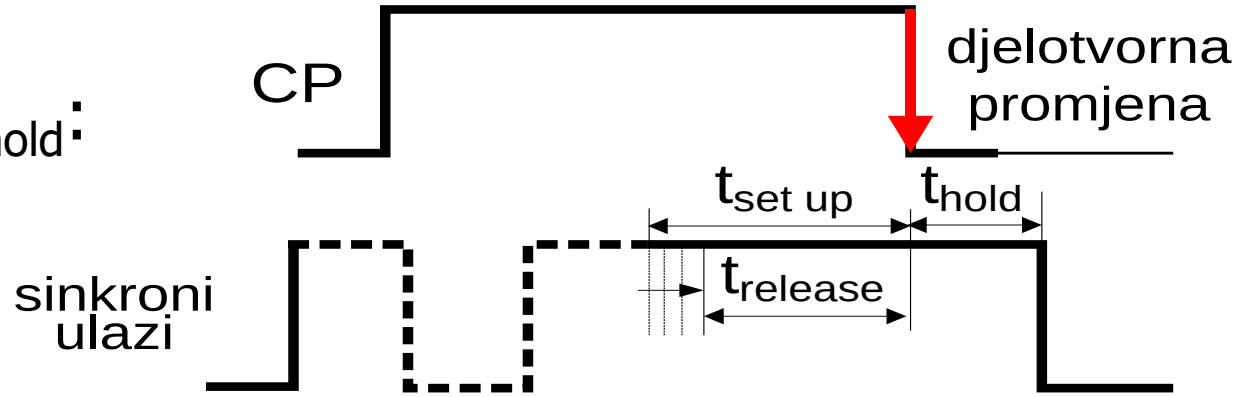
  češće: *vrijeme proleta* (*propagacije*)
  ~ posebno za $0 \rightarrow 1$, odnosno $1 \rightarrow 0$

Vremenski odnosi

- *vrijeme postavljanja* bistabila, $t_{set\ up}$
  ~ *minimalno* vrijeme održavanja podatka
  na *sinkronim* ulazima *prije* djelotvorne promjene CP
  (dvostruki bistabil: prekid veze ulaz-glavni bistabil),
  a da bistabil *sigurno* prihvati podatak
- *vrijeme otpuštanja* (*oslobađanja*) *bistabila*, $t_{release}$
  (analogno $t_{set\ up}$)
  ~ *maksimalno* vrijeme održavanja podatka na *sinkronim*
  ulazima, a da ga bistabil sigurno *ne* prihvati
- *vrijeme pridržavanja* bistabila, t_{hold}
  ~ *minimalno* vrijeme održavanja podatka na sinkronim
  ulazima *nakon* djelotvorne promjene CP;
  potrebno kod nekih izvedbi bistabila

Vremenski odnosi

- definicija $t_{set\ up}$, $t_{release}$, t_{hold}:



- tipični parametri za TTL bistabile serije 74 (t_{PLH} i t_{PHL} za sinkrone ulaze):

	bridom okidani	dvostruki
	7474	**7472**
f_{max} [MHz]	25	20
t_{PLH} [ns]	14	16
t_{PHL} [ns]	20	25
$t_{set\ up}$ [ns]	20	0
t_{hold} [ns]	5	0

Vremenski odnosi

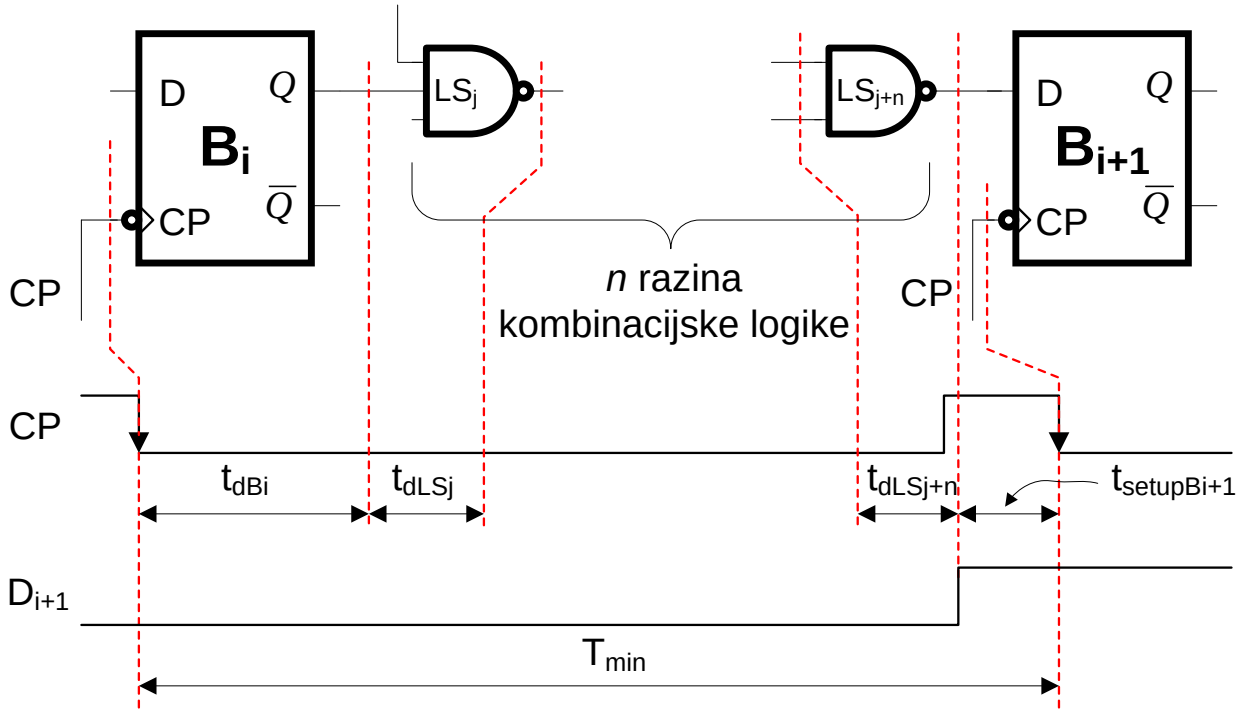
- vremenski odnosi u sekvencijskom sklopu
  ~ *dvije* značajne veličine:
  - maksimalna frekvencija rada sekvencijskog sklopa
    ~ najveća frekvencija CP,
    a da sklop (= svi njegovi bistabili)
    sigurno mijenja stanje kad to ulazi zahtijevaju
  - raskorak, razdešenost ritma
    ~ najveće dozvoljeno *kašnjenje okidanja*
    različitih bistabila u sklopu,
    a da sklop *sigurno mijenja stanje*
    kad to ulazi zahtijevaju

Maksimalna frekvencija rada

- *maksimalna frekvencija* rada sekvencijskog sklopa:
  - težnja
    ~ što viša f_{max}
    - veći broj operacija/sec
    - veća brzina rada digitalnog sklopa
  - problem
    ~ kašnjenje signala
    na stazi između *dva susjedna* stupnja ($\rightarrow$ bistabila)
  - odrediti *najmanji* $T_{min} = 1/f_{max}$ koji osigurava ispravno okidanje bistabila *slijedećeg stupnja*,
    za *najlošiju* stazu signala
    ~ *najviše* kombinacijske logike između bistabila

Maksimalna frekvencija rada

- maksimalna frekvencija, f_{max}:



$$T_{\min} = t_{db}\big|_{\max} + n \cdot t_{dLS}\big|_{\max} + t_{setup}\big|_{\max} \Rightarrow f_{\max} = \frac{1}{T_{\min}}$$

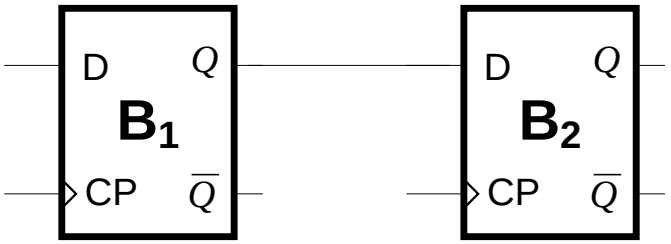
Raskorak

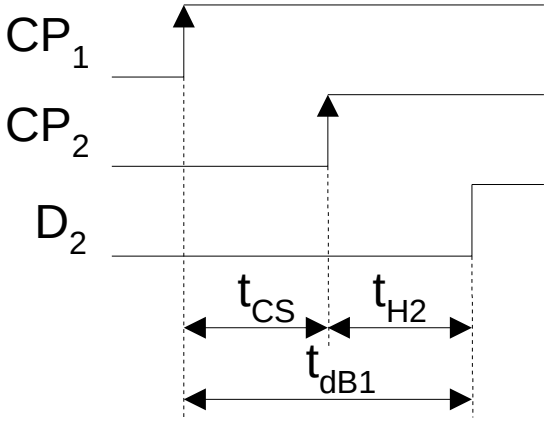
- *raskorak*, *razdešenost ritma* (engl. clock skew):
  - takt (pobuda radi promjene stanja)
    ~ tipično iz *jedinstvenog* generatora
  - problem s *istovremenim* okidanjem bistabila:
    - *različite* duljine vodova do bistabila
    - opterećenje *jednog* generatora signala CP obično je preveliko:
      - signal CP iz jednog generatora distribuirati putem *više* pojačala
        ~ *različita kašnjenja* pojedinih pojačala
      - koristiti *više* generatora signala CP
        ~ *različita kašnjenja* pojedinih generatora

Raskorak

Primjer: neistovremeno okidanje bistabila

- osigurati ispravan upis u B_2 *prethodnog* stanja B_1
- novo stanje B_1 *ne smije se pojaviti* na ulazu B_2 prije nego je B_2 ispravno prihvatio prethodno stanje B_1

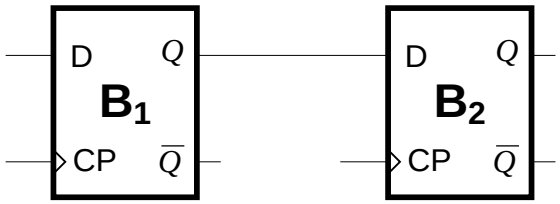




Raskorak

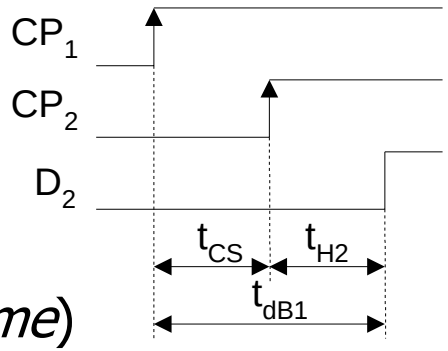
- za najlošiji slučaj vrijedi:

$$(t_{dB1})_{\min} \geq (t_{H2})_{\max} + (t_{CS})_{\max}$$



- osigurati ograničenje razdešenosti ritma:

$$(t_{CS})_{\max} \leq (t_{dB1})_{\min} - (t_{H2})_{\max}$$

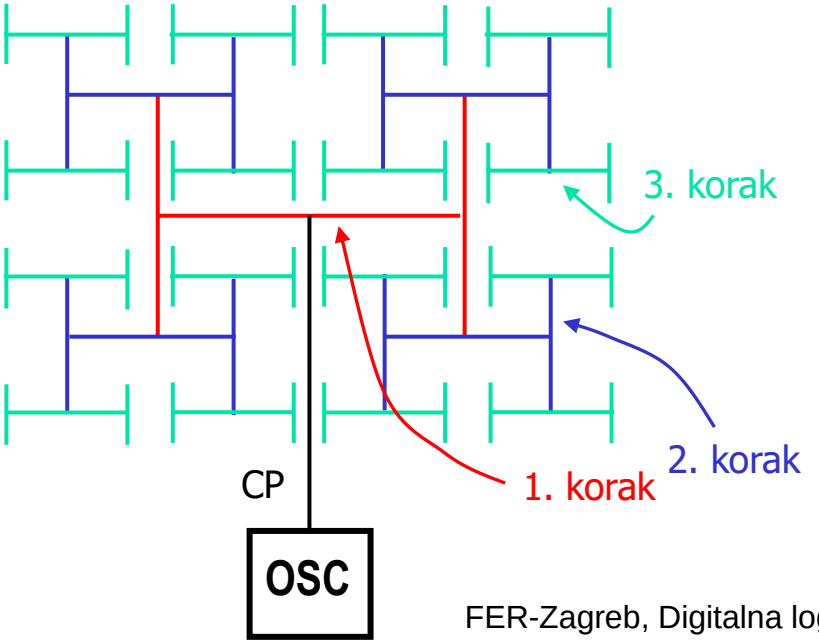


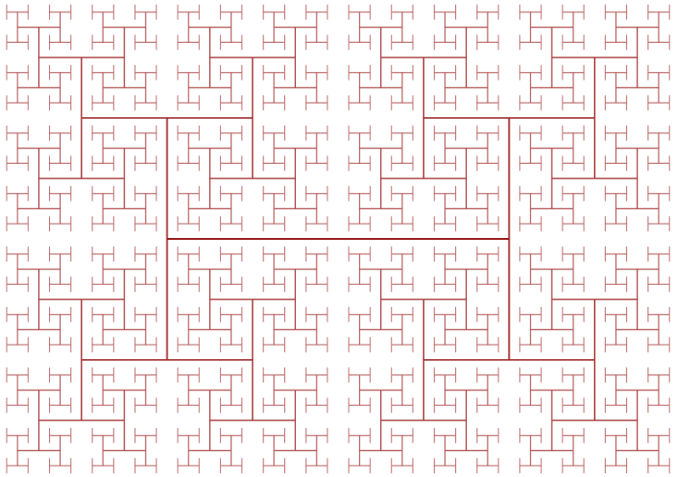
t_{cs} : raskorak (engl. *clock skew time*)

t_H: vrijeme držanja (engl. *hold time*)

Raskorak

- uobičajeno rješenje problema raskoraka u VLSI
  ~ distribucija signala CP mrežom karakteristična oblika, temeljenom na fraktalima ($\rightarrow$ *H stablo*, engl. H tree):
  - odnos poprečnih segmenata: $1 : \sqrt{2}$
  - podjednako kašnjenje signala takta





Literatura

U. Peruško, V. Glavinić: *Digitalni sustavi*, Poglavlje 8: Postupak projektiranja s osvrtom na jezik VHDL; Poglavlje 9: Sinkroni sekvencijski sklopovi. Poglavlje 5: Bistabil.

- konceptualizacija sekvencijskih sklopova: str. 335-341
- projektiranje sekvencijskih sklopova: str. 354-375
- izvedbe sekvencijskih sklopova: str. 379-380
- analiza sekvencijskih sklopova: str. 341-354
- vremenski odnosi: str. 376-379
- modeliranje sekvencijskih sklopova u jeziku VHDL: str. 327-330
- karakteristični dinamički parametri: str. 195-196

Zadaci za vježbu (1)

U. Peruško, V. Glavinić: *Digitalni sustavi*, Poglavlje 9: Sinkroni sekvencijski sklopovi; Poglavlje 11: Sekvencijski moduli: registri i brojila.

- projektiranje sekvencijskih sklopova: 9.5-9.9, 9.14, 9.17, 9.18, 9.22, 9.23, 9.26
- izvedbe sekvencijskih sklopova: 9.15, 9.16; 11.25
- analiza sekvencijskih sklopova: 9.5-9.9, 9.14, 9.17, 9.18, 9.22, 9.23, 9.26
- modeliranje sekvencijskih sklopova u jeziku VHDL: 9.20

Zadaci za vježbu (2)

M. Čupić: *Digitalna elektronika i digitalna logika. Zbirka riješenih zadataka*, Cjelina 11: Strojevi s konačnim brojem stanja.

- konceptualizacija sekvencijskih sklopova:
  - riješeni zadaci: 11.6, 11.7
- projektiranje sekvencijskih sklopova:
  - riješeni zadaci: 11.1-11.5, 11.8, 11.9, 11.12, 11.13, 11.15
  - zadaci za vježbu: 6, 7
- izvedbe sekvencijskih sklopova:
  - riješeni zadaci: 11.14
  - zadaci za vježbu: 4
- modeliranje sekvencijskih sklopova u jeziku VHDL:
  - riješeni zadaci: 11.18-11.23
  - zadaci za vježbu: 2, 3, 8