



# 10. Sekvencijski sklopovi

---



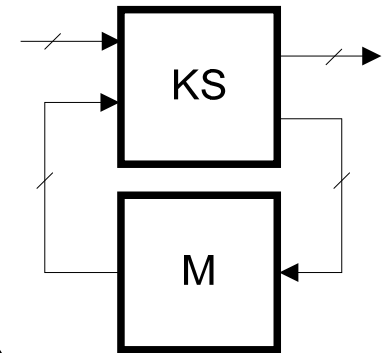
# Sadržaj predavanja

---

- **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  
 $\sim$  *strojevi stanja* (engl. state machines)
  - stanja je konačno mnogo ( $2^n$ )  
 $\sim$  strojevi *s konačnim brojem stanja*  
(engl. finite state machines)
  - slijed operacija u sekvencijskom sklopu  
 $\sim$  "ugrađeni" algoritam:  
*algoritamski stroj stanja*
  - operacije se obavljaju bez čovjekove pomoći  
 $\sim$  *automat*:  
*digitalni automat, konačni automat*

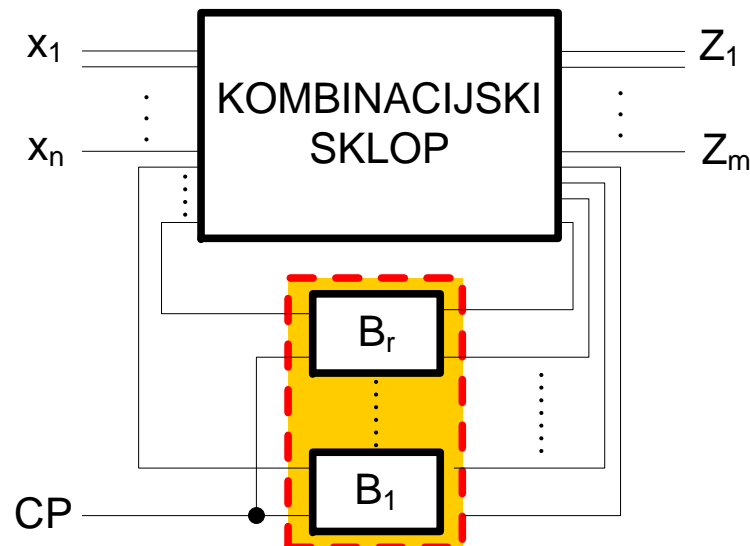
# Sinkroni sekvencijski sklopovi

- 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 *više-bitnih* podataka





# 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

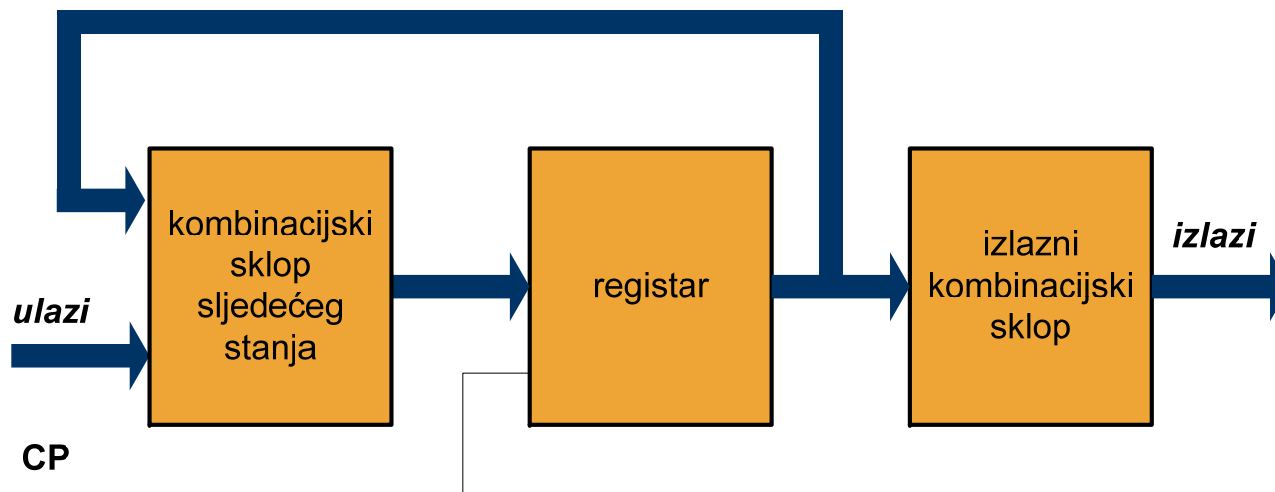
---

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



# Mooreov model

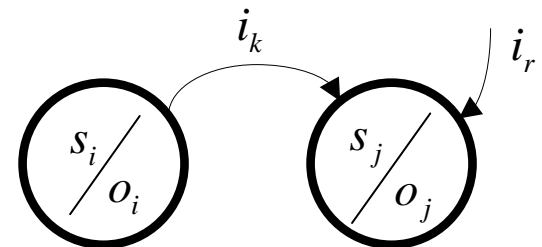
- "automat stanja"  
~ izlaz ovisi *samo o* unutarnjem stanju



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

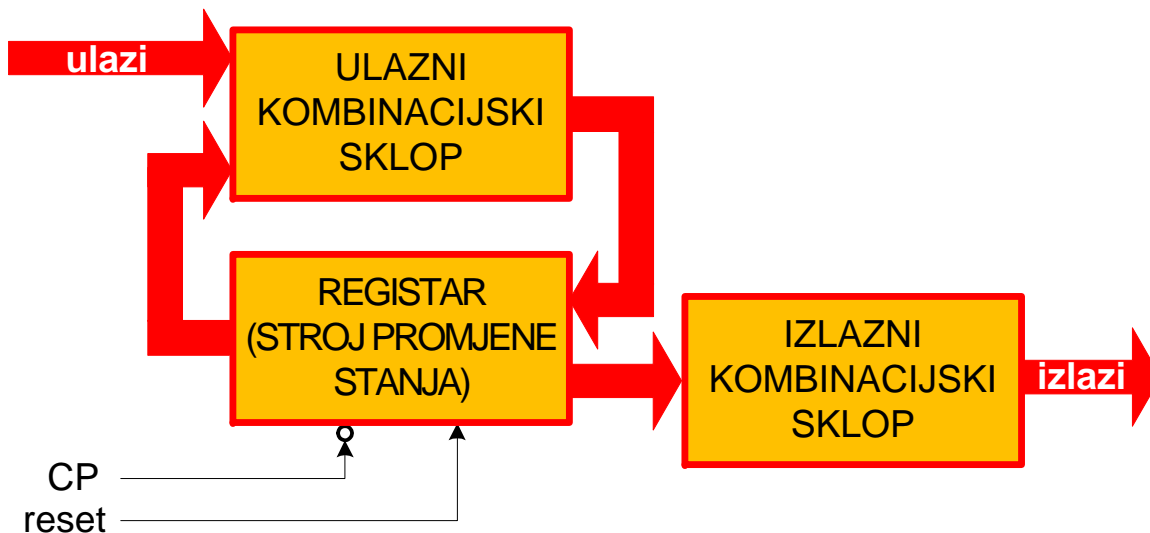
$$\delta: S \times I \rightarrow S$$

$$\mu: S \rightarrow O$$



# Mooreov model

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



# 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 DOWNT0 0);
    CONSTANT S0: std_logic_vector(K DOWNT0 0) := "0...000";
    CONSTANT S1: std_logic_vector(K DOWNT0 0) := "0...001";
    -- 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

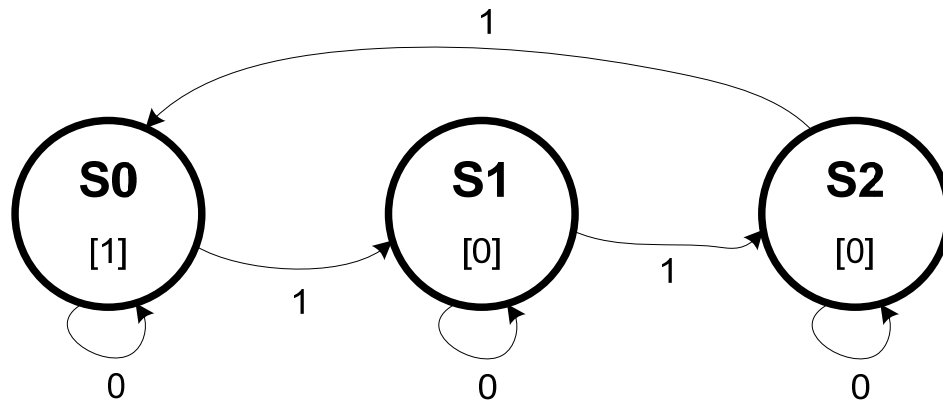
---

- 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;
```

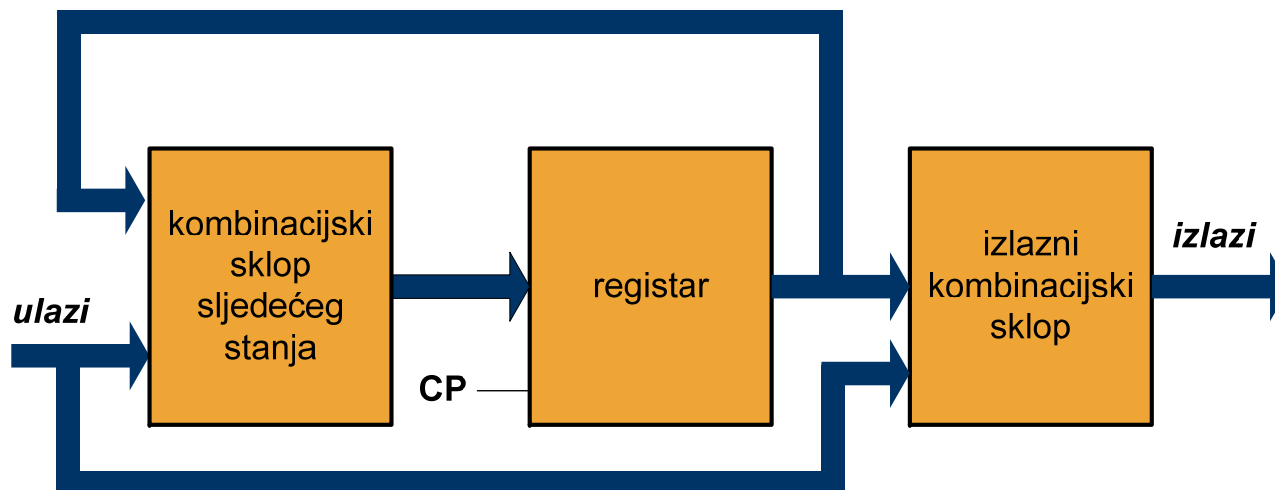
```
END BEHAVIORAL;
```

2

3

# Mealyjev model

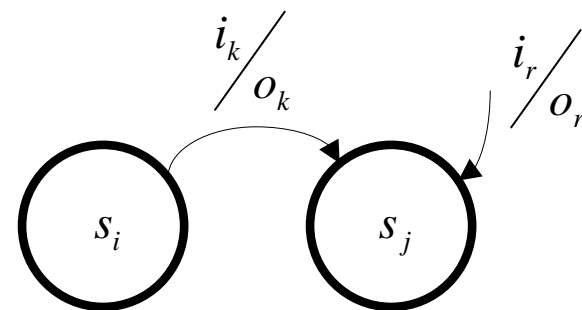
- *Mealyjev model* : "automat prijelaza"  
~ izlaz ovisi o unutarnjem stanju i o ulazu



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

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

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



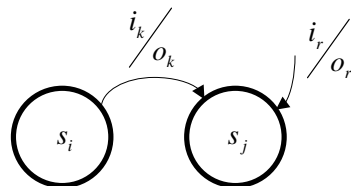
# Mealyjev model

- ekvivalencija Mooreovog i Mealyjevog automata:

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

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

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



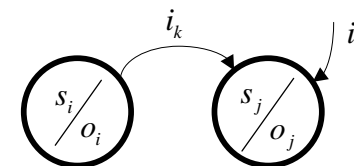
$$q^{n+1} = \delta(q^n, x^n)$$

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

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

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

$$\mu: S \rightarrow O$$



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

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

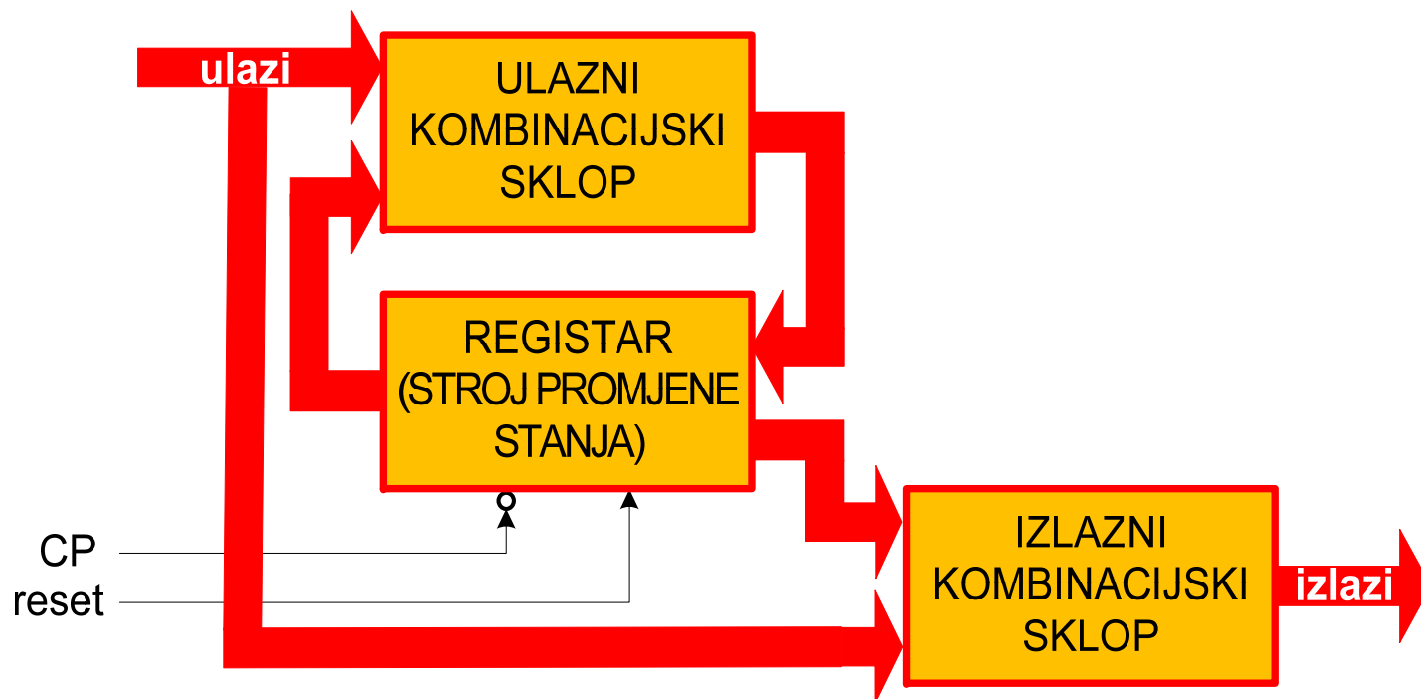
$$= \mu(\delta(q^{n-1}, x^{n-1}))$$

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

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

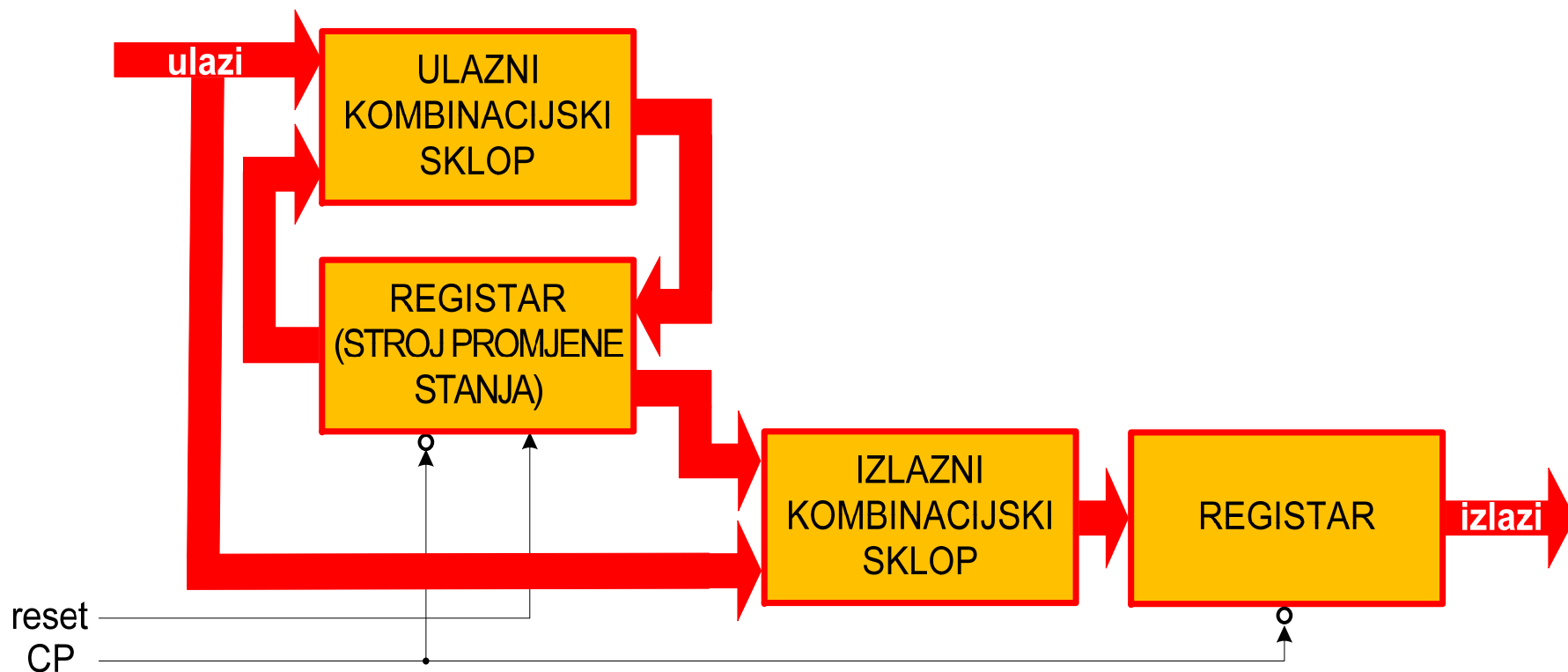
# Mealyjev model

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



# Mealyjev model

- verzija s registrima na izlazu  
(stabilne vrijednosti izlaza između dva susjedna impulsa CP)



# 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 DOWNT0 0);
    reset: IN std_logic;
    izlazi: OUT std_logic_vector(M DOWNT0 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 čiste
    -- osjetljivosti; npr. bezuvjetni prijelaz u stanje S0
    -- i sve '0' na izlazima:
    state_next <= S0;
    izlazi_next <= "000..000"; -- niz od (K+1) nule.
  END PROCESS;
```

1  
2



# Mealyjev model

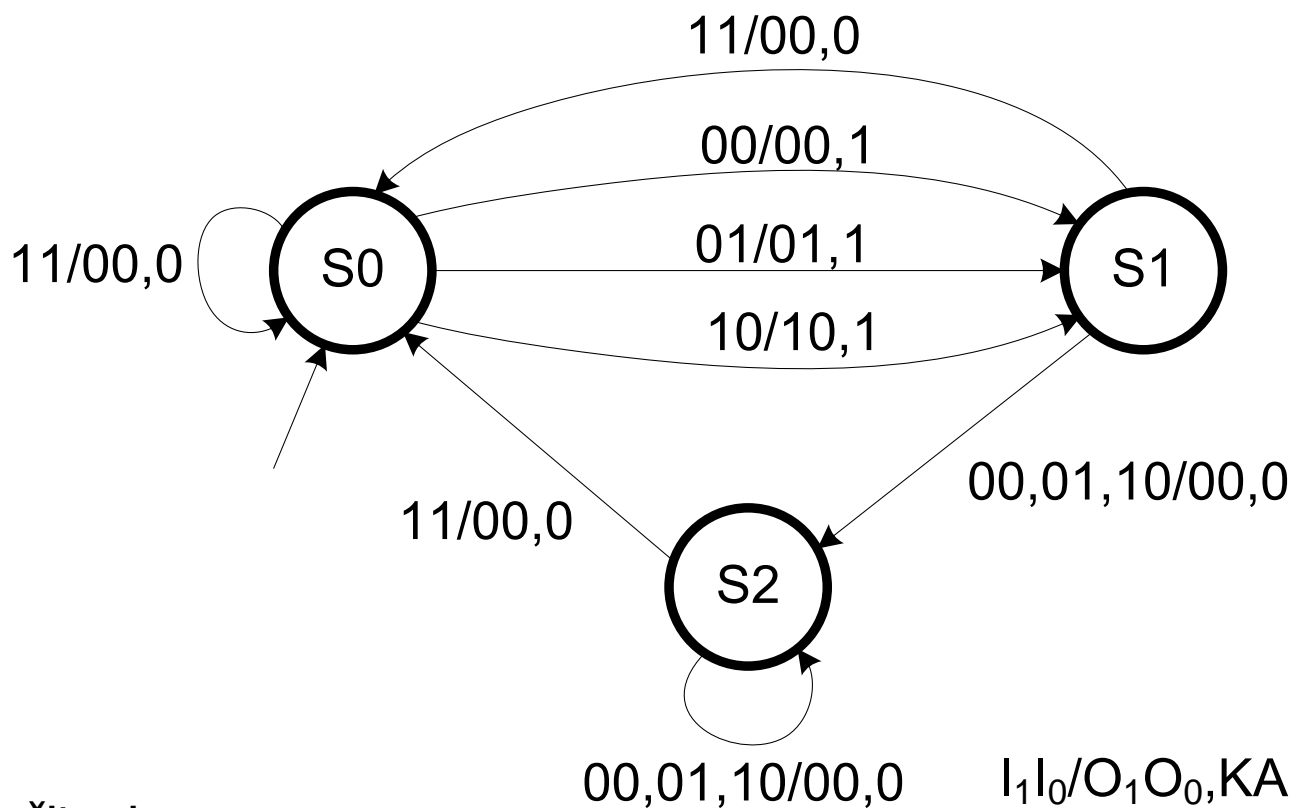
```
-- stroj promjene stanja i registar; promjena stanja i izlaza
-- na temelju signala clock i asinkronih ulaza
PROCESS( clock, reset )
BEGIN
    -- asinkroni ulaz(i) dominira(ju)
    IF reset = '1' THEN
        state_present <= S0;  -- postaviti stanje u S0
        izlaz <= "000...00";  -- postaviti 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





# 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;
    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

---

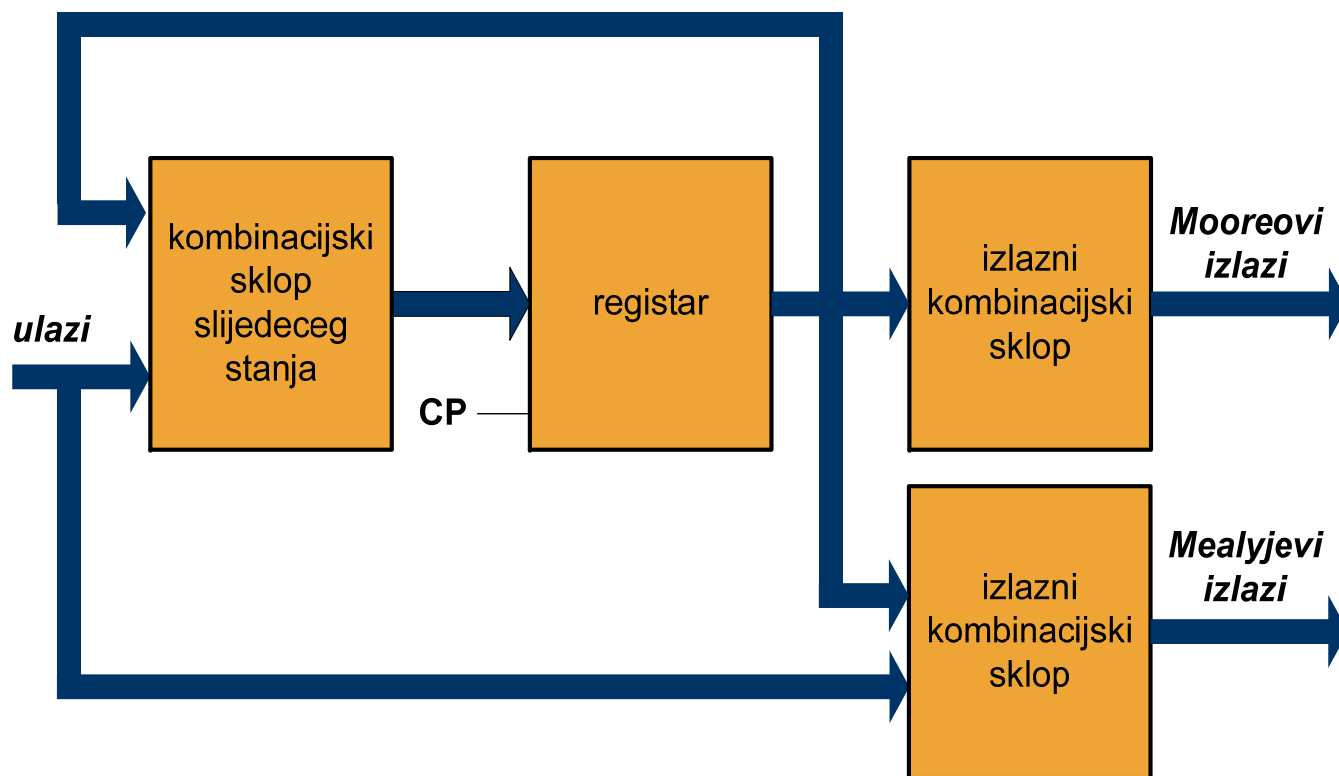
3

```
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;
```

# 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  $2^{n-1} < m \leq 2^n$
  3. minimiziranje broja stanja  
~ 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* jednažbi bistabila  
~ uzbuda potrebna za odgovarajući prijelaz  
(→ generiranje slijedećeg stanja)
    - dobivanje *izlaznih* jednažbi 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*:
    - za *potpuno specificirane* sklopove
      - ~  $\forall$  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 ta stanja  
~ automat s *reduciranim* brojem unutarnjih stanja  
⇒ *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

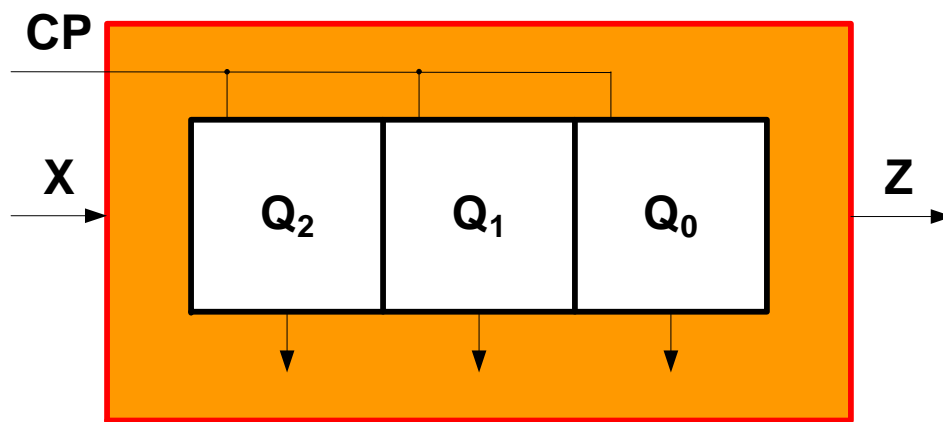
# Minimiziranje memorije

- 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  
~ klasa naravno ima *manje* od stanja 😊

# Minimiziranje memorije

*Primjer:* Huffman-Mealyjeva metoda

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

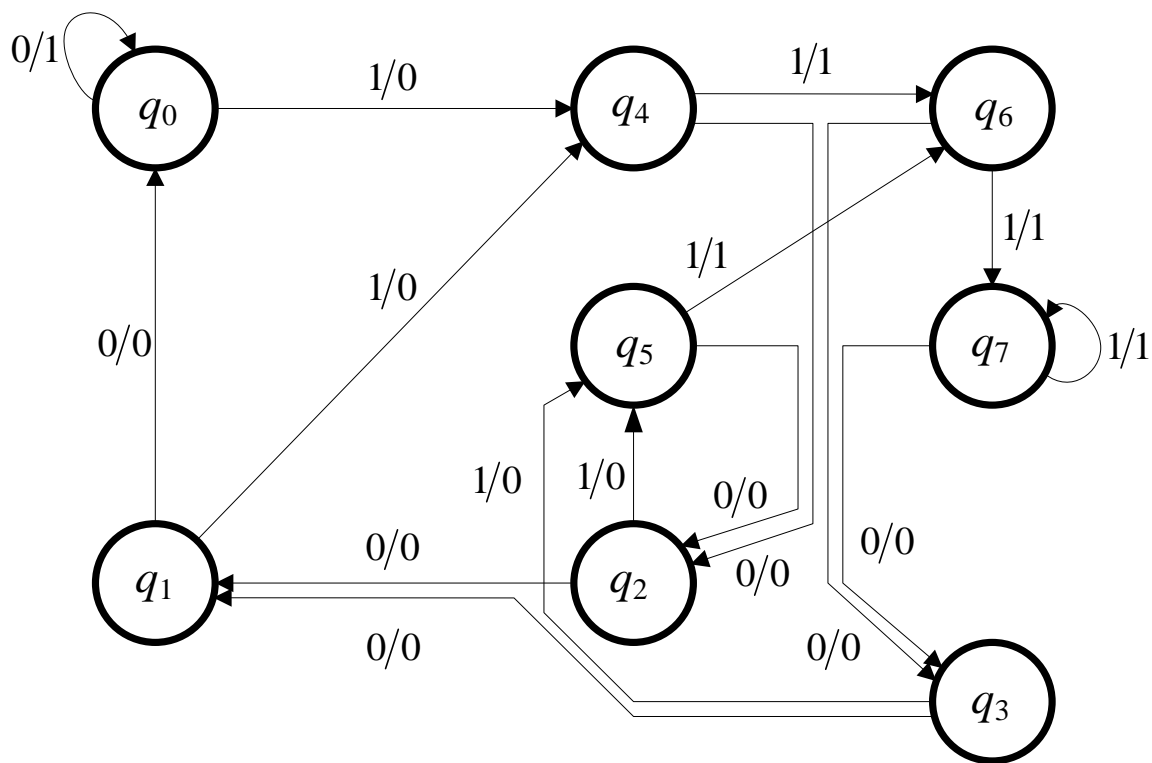


| $q^n$ | $q^{n+1}, z^n$ |          |
|-------|----------------|----------|
|       | $x^n=0$        | $x^n=1$  |
| $q_0$ | $q_0, 1$       | $q_4, 0$ |
| $q_1$ | $q_0, 0$       | $q_4, 0$ |
| $q_2$ | $q_1, 0$       | $q_5, 0$ |
| $q_3$ | $q_1, 0$       | $q_5, 0$ |
| $q_4$ | $q_2, 0$       | $q_6, 1$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

# 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

$$a = \{q_0\}$$

$$b = \{q_1, q_2, q_3\}$$

$$c = \{q_4, q_5, q_6, q_7\}$$

- provjera prijelaza  
~ 4 klase ekvivalentnih stanja

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

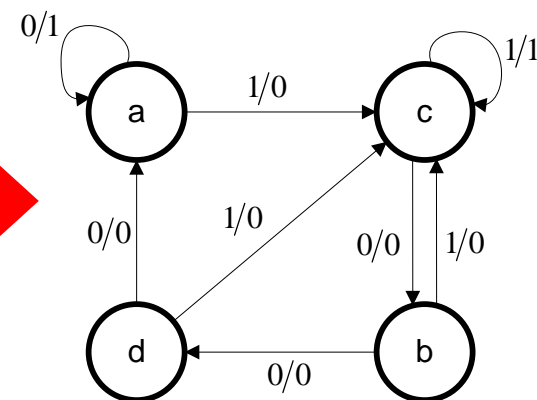
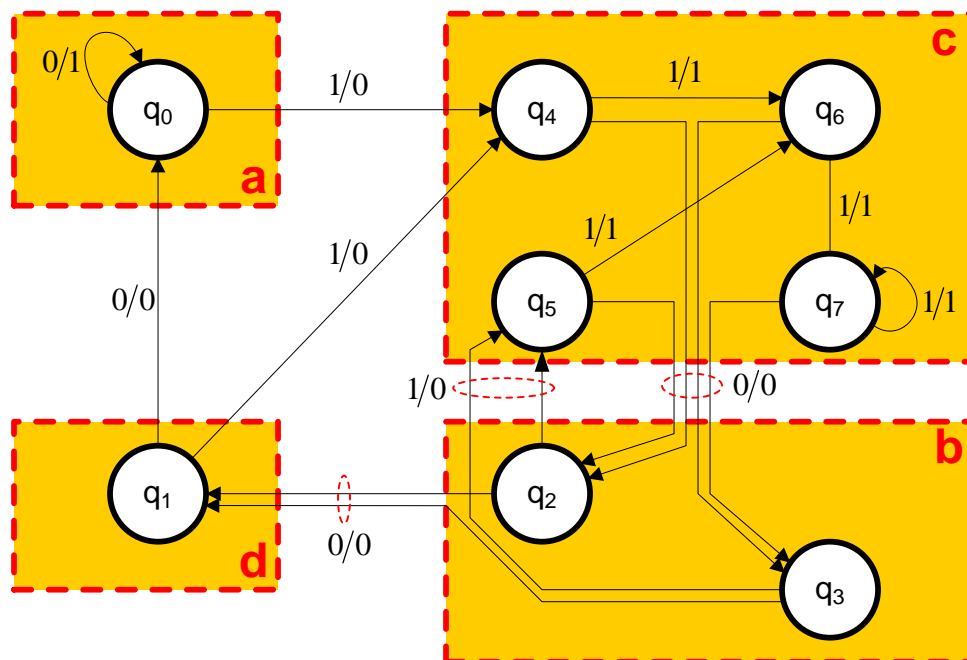
| klasa     | a     | b     |       |       | c     |       |       |       |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| stanje    | $q_0$ | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ |
| sl. klasa | a c   | a c   | b c   | b c   | b c   | b c   | b c   | 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<br>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       |



| $(B_1B_0)^n$ | $(B_1B_0)^{n+1}$ |       | $z^n$ |       |
|--------------|------------------|-------|-------|-------|
|              | $x=0$            | $x=1$ | $x=0$ | $x=1$ |
| 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     |



$B_1^{n+1}$

|     | $B_1B_0$ |    |    |    |
|-----|----------|----|----|----|
|     | 00       | 01 | 11 | 10 |
| X 0 |          | 1  |    |    |
| 1   | 1        | 1  | 1  | 1  |

$B_0^{n+1}$

|     | $B_1B_0$ |    |    |    |
|-----|----------|----|----|----|
|     | 00       | 01 | 11 | 10 |
| X 0 |          | 1  |    | 1  |
| 1   |          |    |    |    |

$Z$

|     | $B_1B_0$ |    |    |    |
|-----|----------|----|----|----|
|     | 00       | 01 | 11 | 10 |
| X 0 | 1        |    |    |    |
| 1   |          |    |    | 1  |

# Ostvarivanje sekvencijskog sklopa

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

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

---

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

$$\begin{aligned} D_0 &= \overline{X} \cdot (\overline{B}_1 B_0 + B_1 \overline{B}_0) \\ &= \overline{X} \cdot (B_1 \oplus B_0) \end{aligned}$$

$$Z = \overline{B}_1 \overline{B}_0 \overline{X} + B_1 \overline{B}_0 X$$

|     |   | $B_1 B_0$ |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 |           | 1  |    |    |
|     | 1 | 1         | 1  | 1  | 1  |

|     |   | $B_1 B_0$ |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 |           | 1  |    | 1  |
|     | 1 |           |    |    |    |

|     |   | $B_1 B_0$ |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 | 1         |    |    |    |
|     | 1 |           |    |    | 1  |

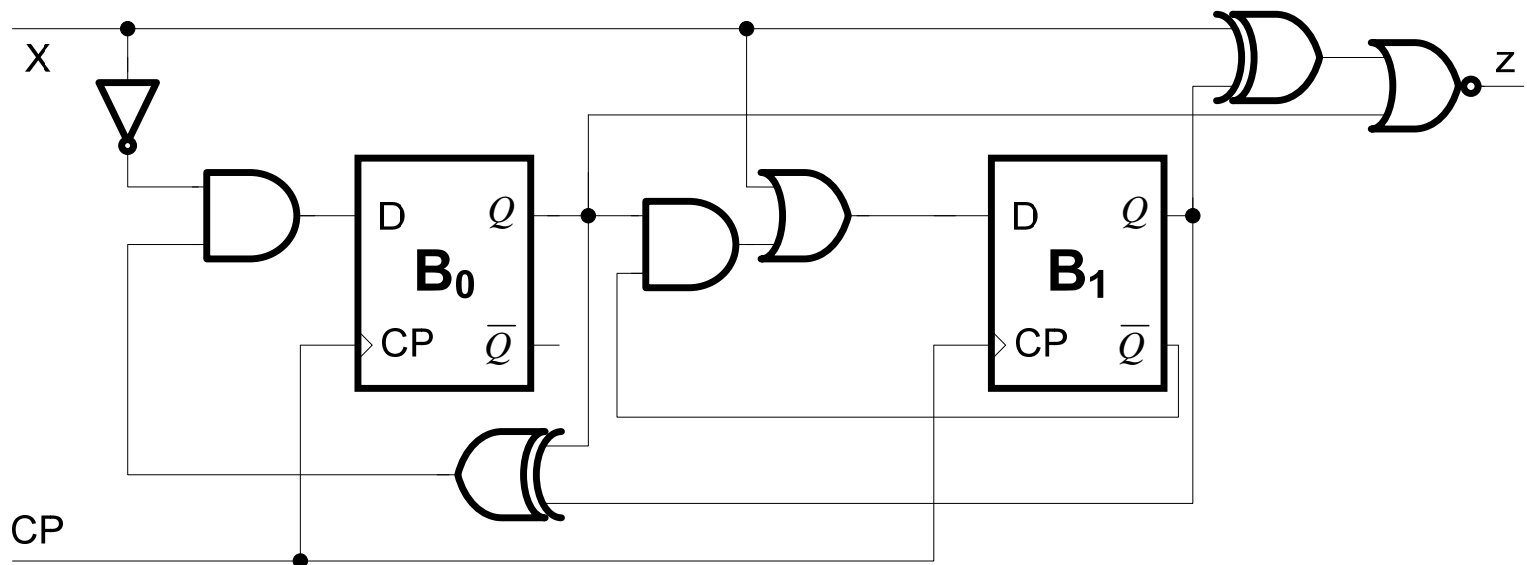
# Ostvarivanje sekvencijskog sklopa

- logička shema:

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

$$\begin{aligned} D_0 &= \bar{X} \cdot (\bar{B}_1 B_0 + B_1 \bar{B}_0) \\ &= \bar{X} \cdot (B_1 \oplus B_0) \end{aligned}$$

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



# Ostvarivanje sekvencijskog sklopa

*Primjer:* izvedba memorije JK-bistabilima

- koristi se *uzbudna tablica*:

|                             | $Q^n$ | $Q^{n+1}$ | $J$ | $K$ |
|-----------------------------|-------|-----------|-----|-----|
| $\alpha := 0 \rightarrow 1$ |       |           | 1   | x   |
| $\beta := 1 \rightarrow 0$  |       |           | x   | 1   |
| $0 := 0 \rightarrow 0$      |       |           | 0   | x   |
| $1 := 1 \rightarrow 1$      |       |           | x   | 0   |

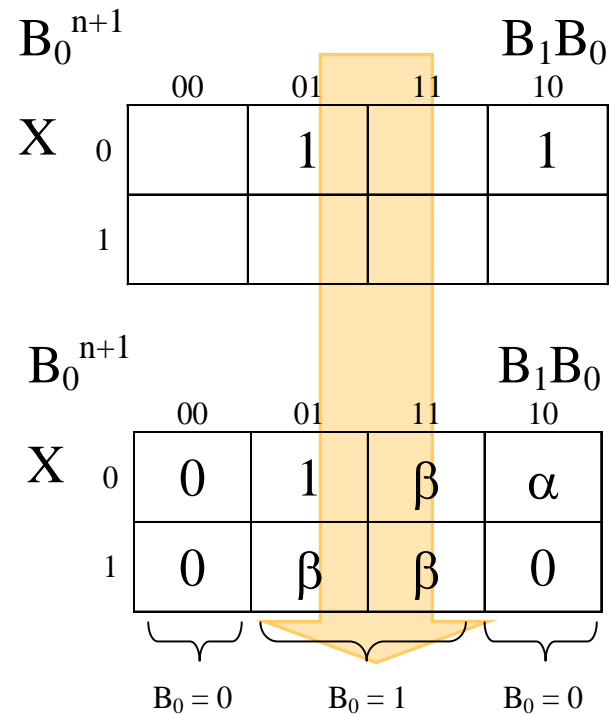
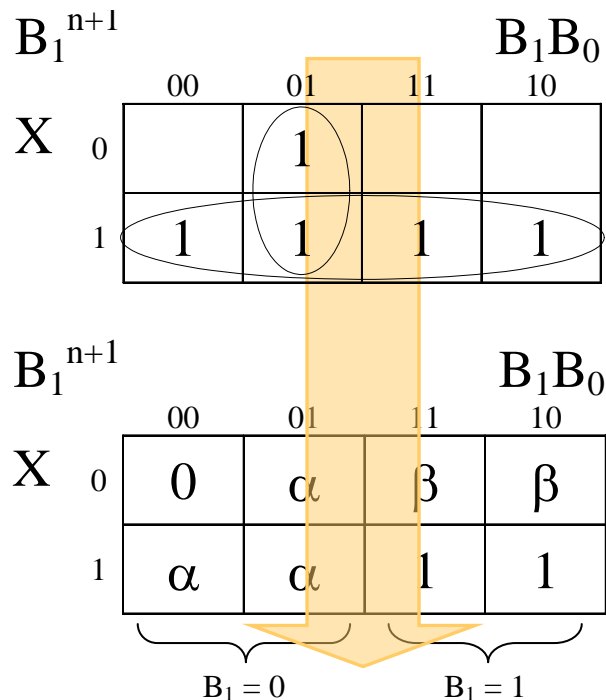
oznaka vrste prijelaza      vrsta prijelaza      potrebna uzbuda JK-bistabila



# Ostvarivanje sekvencijskog sklopa

- označavanje prijelaza zapisom *uzbudne tablice*:

| $Q^n$            | $Q^{n+1}$ | J | K |
|------------------|-----------|---|---|
| 0 → 1 : $\alpha$ |           | 1 | x |
| 1 → 0 : $\beta$  |           | x | 1 |
| 0 → 0 : 0        |           | 0 | x |
| 1 → 1 : 1        |           | x | 0 |



# Ostvarivanje sekvencijskog sklopa

- izvođenje ulaznih jednažbi:

| $Q^n$ | $Q^{n+1}$ |            | J | K |
|-------|-----------|------------|---|---|
| 0     | → 1       | : $\alpha$ | 1 | x |
| 1     | → 0       | : $\beta$  | x | 1 |
| 0     | → 0       | : 0        | 0 | x |
| 1     | → 1       | : 1        | x | 0 |

$B_1^{n+1}$

|       |          |          |         |         |
|-------|----------|----------|---------|---------|
|       | 00       | 01       | 11      | 10      |
| $X$ 0 | 0        | $\alpha$ | $\beta$ | $\beta$ |
| 1     | $\alpha$ | $\alpha$ | 1       | 1       |

$B_1 B_0$

$B_1 = 0$        $B_1 = 1$

$B_0^{n+1}$

|       |    |         |         |          |
|-------|----|---------|---------|----------|
|       | 00 | 01      | 11      | 10       |
| $X$ 0 | 0  | 1       | $\beta$ | $\alpha$ |
| 1     | 0  | $\beta$ | $\beta$ | 0        |

$B_1 B_0$

$B_0 = 0$        $B_0 = 1$        $B_0 = 0$

$J_1$

|       |    |    |    |    |
|-------|----|----|----|----|
|       | 00 | 01 | 11 | 10 |
| $X$ 0 |    | 1  | x  | x  |
| 1     | 1  | 1  | x  | x  |

$B_1 B_0$

$J_0$

|       |    |    |    |    |
|-------|----|----|----|----|
|       | 00 | 01 | 11 | 10 |
| $X$ 0 |    | x  | x  | 1  |
| 1     |    | x  | x  |    |

$B_1 B_0$

$K_1$

|       |    |    |    |    |
|-------|----|----|----|----|
|       | 00 | 01 | 11 | 10 |
| $X$ 0 | x  | x  | 1  | 1  |
| 1     | x  | x  |    |    |

$B_1 B_0$

$K_0$

|       |    |    |    |    |
|-------|----|----|----|----|
|       | 00 | 01 | 11 | 10 |
| $X$ 0 | x  |    | 1  | x  |
| 1     | x  | 1  | 1  | x  |

$B_1 B_0$

# Ostvarivanje sekvencijskog sklopa

- ulazne jednačbe:

$J_1$

|     |   |           |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | $B_1 B_0$ |    |    |    |
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 |           | 1  | x  | x  |
|     | 1 | 1         | 1  | x  | x  |

$J_0$

|     |   |           |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | $B_1 B_0$ |    |    |    |
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 |           | x  | x  | 1  |
|     | 1 |           | x  | x  |    |

$K_1$

|     |   |           |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | $B_1 B_0$ |    |    |    |
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 | x         | x  | 1  | 1  |
|     | 1 | x         | x  |    |    |

$K_0$

|     |   |           |    |    |    |
|-----|---|-----------|----|----|----|
|     |   | $B_1 B_0$ |    |    |    |
|     |   | 00        | 01 | 11 | 10 |
| $X$ | 0 | x         |    | 1  | x  |
|     | 1 | x         | 1  | 1  | x  |

$$J_1 = B_0 + X$$

$$K_1 = \bar{X}$$

$$J_0 = B_1 \bar{X}$$

$$K_0 = X + B_1$$

# Ostvarivanje sekvencijskog sklopa

- logička shema:

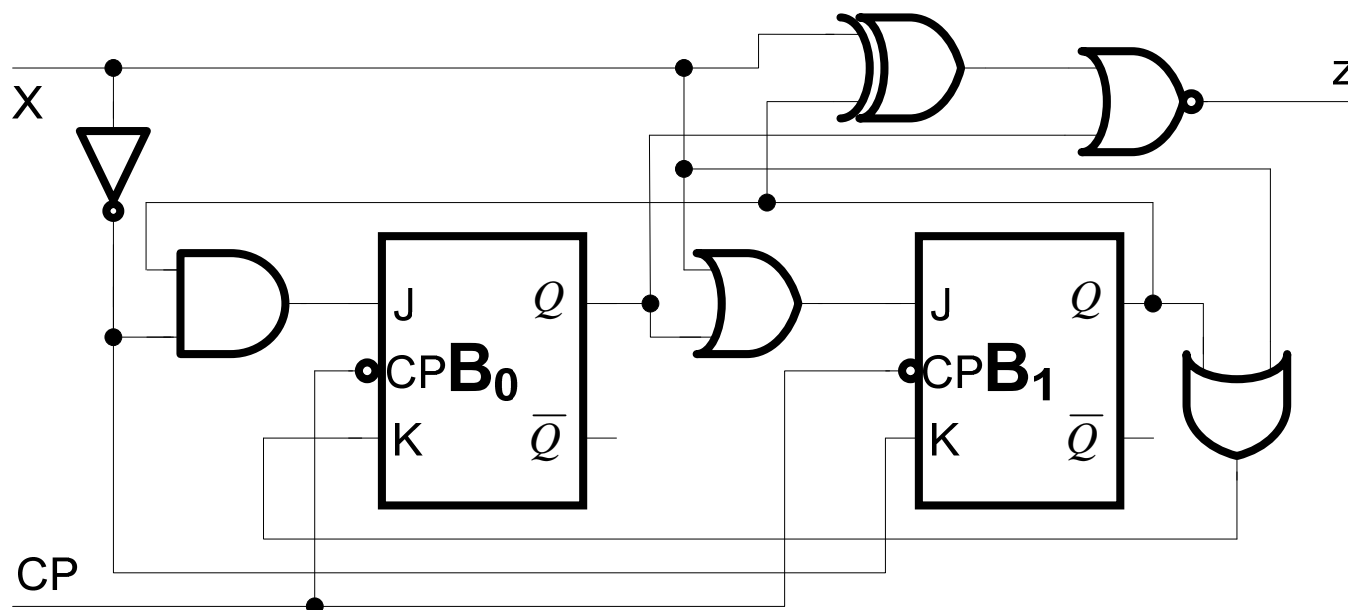
$$J_0 = B_1 \bar{X}$$

$$J_1 = B_0 + X$$

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

$$K_0 = X + B_1$$

$$K_1 = \bar{X}$$





# 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 + kombinacijska logika (osnovni logički sklopovi)  
~ kao prije
  - registar (= niz D bistabila!) + ROM
  - registar + SPLD (npr. PLA, PAL, ...)
  - druga "univerzalna logika":
    - CPLD
    - FPGA
  - sekvencijski moduli



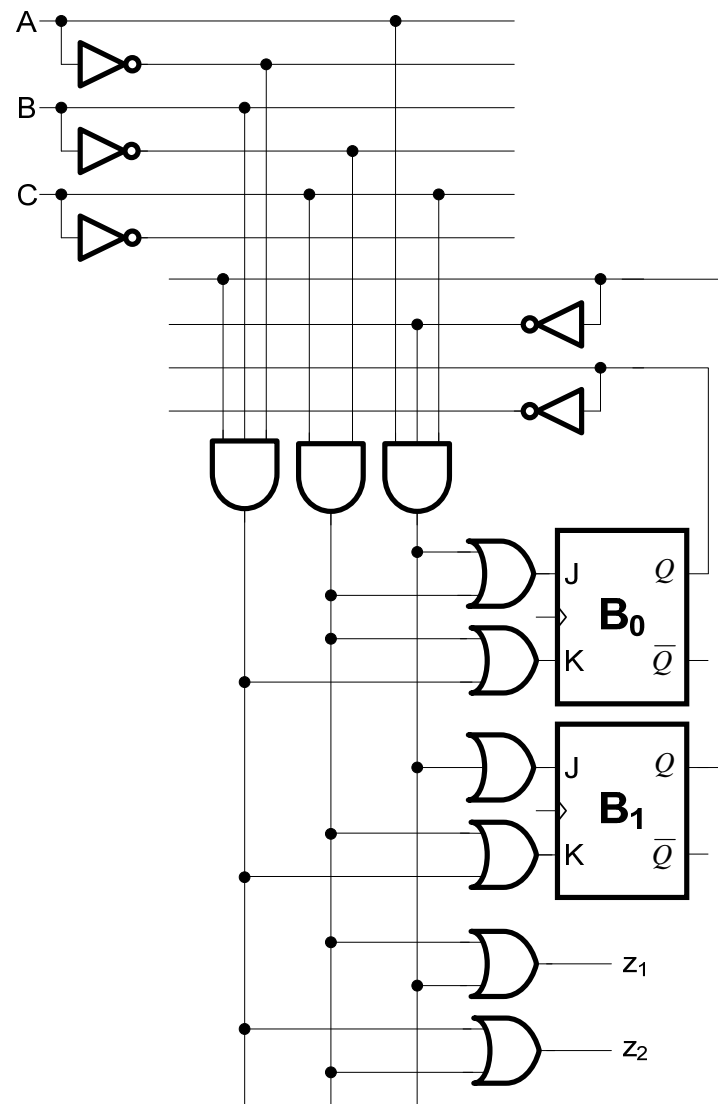
# Izvedbe sekvencijskih sklopova

---

- *sekvencijski moduli*:  
~ cjeline koje sadrže kombinacijski sklop / memoriju  
(niz/skup bistabila ili registara)
- općenita klasifikacija:
  - *standardni* moduli:  $n$ -bitni
    - za funkcije tipa brojanja: npr. brojila
    - za funkcije tipa pohranjivanja podataka:  
npr. registri
  - *univerzalni* moduli  
~ ostvarivanje proizvoljnih sekvencijskih sklopova  
(usp. generiranje 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:

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

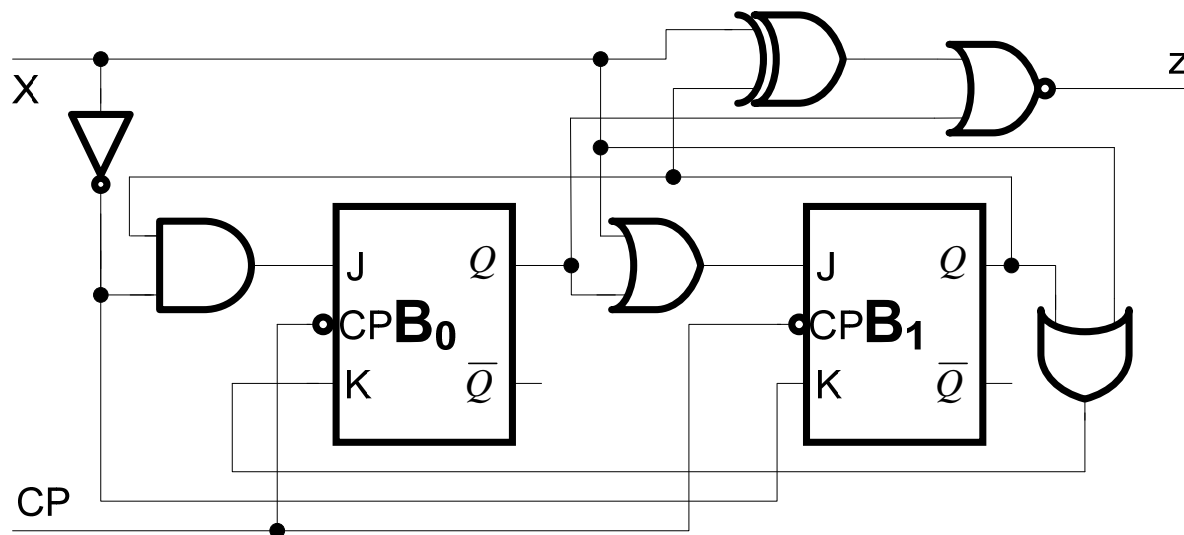


# Analiza sekvencijskih sklopova

- 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 (000...0)

# Analiza sekvencijskih sklopova

*Primjer:* analiza prethodnog sklopa



iz sheme sklopa očitano:

$$J_1 = B_0 + X \quad J_0 = B_1 \bar{X}$$

$$K_1 = \bar{X} \quad K_0 = X + B_1$$

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

# Analiza sekvencijskih sklopova

$$J_1 = B_0 + X$$

$$J_0 = B_1 \bar{X}$$

$$Z = \bar{B}_1 \bar{B}_0 \bar{X} + B_1 \bar{B}_0 X$$

$$K_1 = \bar{X}$$

$$K_0 = X + B_1$$

- pridruživanje:

| B <sub>1</sub> | B <sub>0</sub> | 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 <sub>1</sub> | B <sub>0</sub> | X | J <sub>1</sub> | K <sub>1</sub> | J <sub>0</sub> | K <sub>0</sub> | B <sub>1</sub> | B <sub>0</sub> | 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**
  - maksimalna frekvencija rada
  - raskorak



# 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*  
bistabila u sklopu,  
a da sklop *sigurno mijenja stanje*  
kad to ulazi zahtijevaju

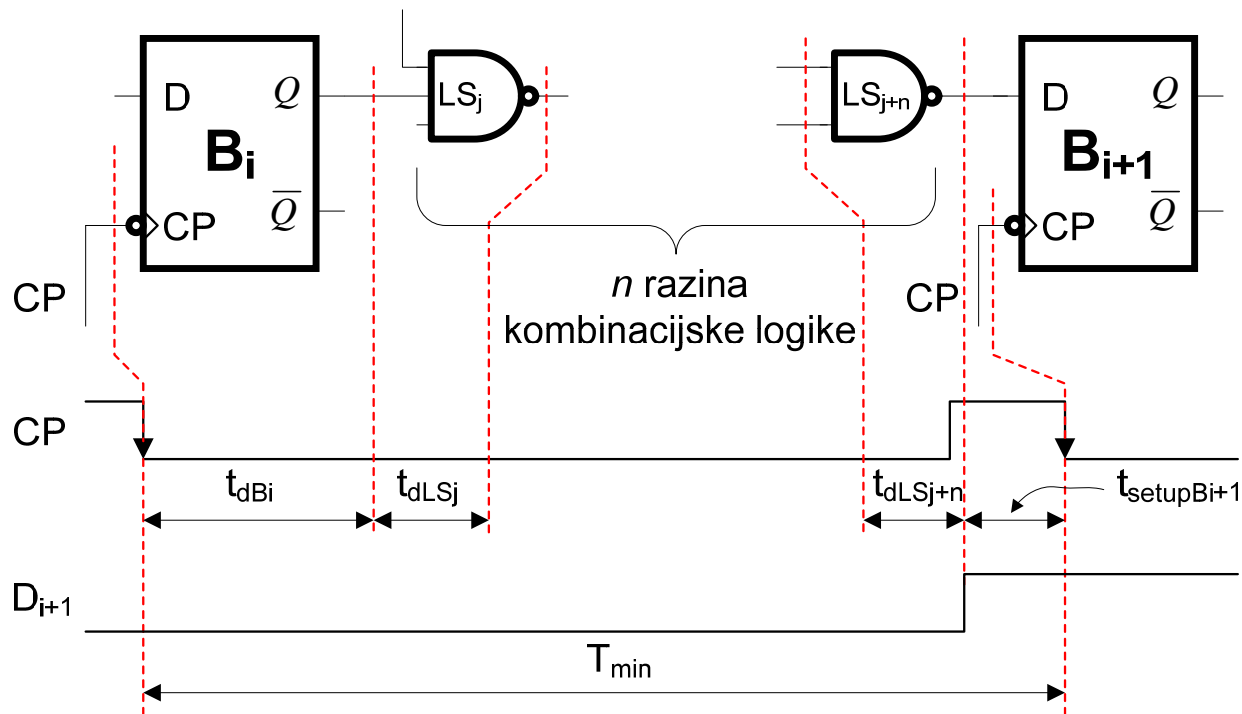
# Maksimalna frekvencija rada

- *maksimalna frekvencija* rada sekvencijskog sklopa:
  - težnja  
~ što viša  $f_{\max}$ 
    - veći broj operacija/sek
    - 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 *sljedećeg stupnja*,  
za *najlošiju* stazu signala  
~ *najviše* kombinacijske logike između bistabila



# Maksimalna frekvencija rada

- maksimalna frekvencija,  $f_{\max}$ :

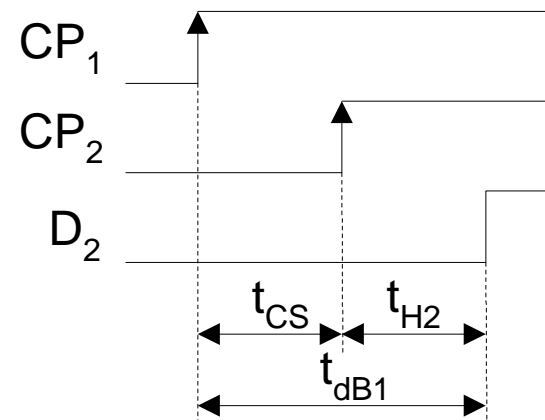
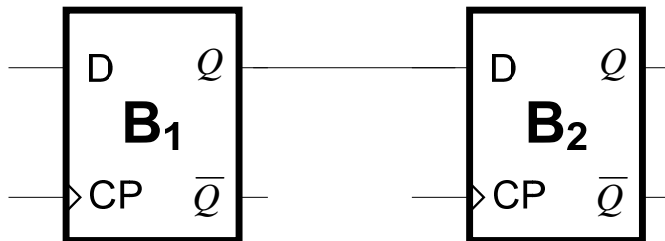


$$T_{\min} = t_{db} \Big|_{\max} + n \cdot t_{dLS} \Big|_{\max} + t_{setup} \Big|_{\max} \Rightarrow f_{\max} = \frac{1}{T_{\min}}$$

- *raskorak, razdešenost ritma* (engl. clock skew):
  - takt (pobuda radi promjene stanja): tipično iz *jedinstvenog* generatora
  - problem s *istovremenim* okidanjem bistabila
    - *različite* duljine vodova do bistabila
    - preveliko ukupno *opterećenje* pobudnih sklopova  
~ signal CP se razvodi iz *više* pobudnih sklopova
      - *više* pojačala
      - *različita kašnjenja* pojedinih pojačala

## *Primjer:* neistovremeno okidanje bistabila

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



- za najlošiji slučaj vrijedi:

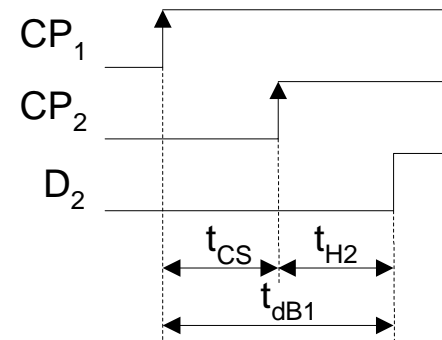
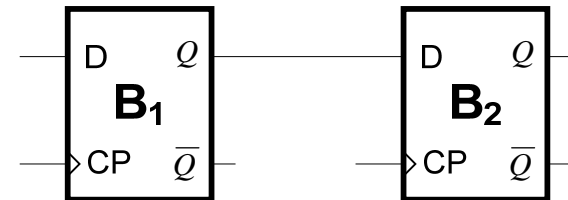
$$(t_{dB1})_{\min} \geq (t_{H2})_{\max} + (t_{CS})_{\max}$$

- osigurati ograničenje razdešenosti ritma:

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

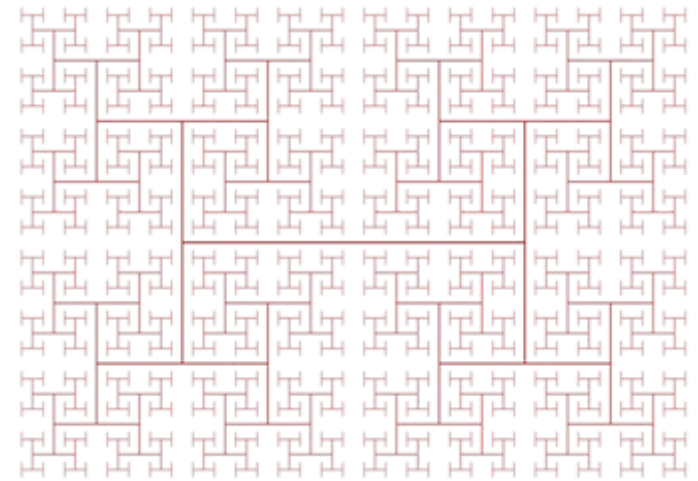
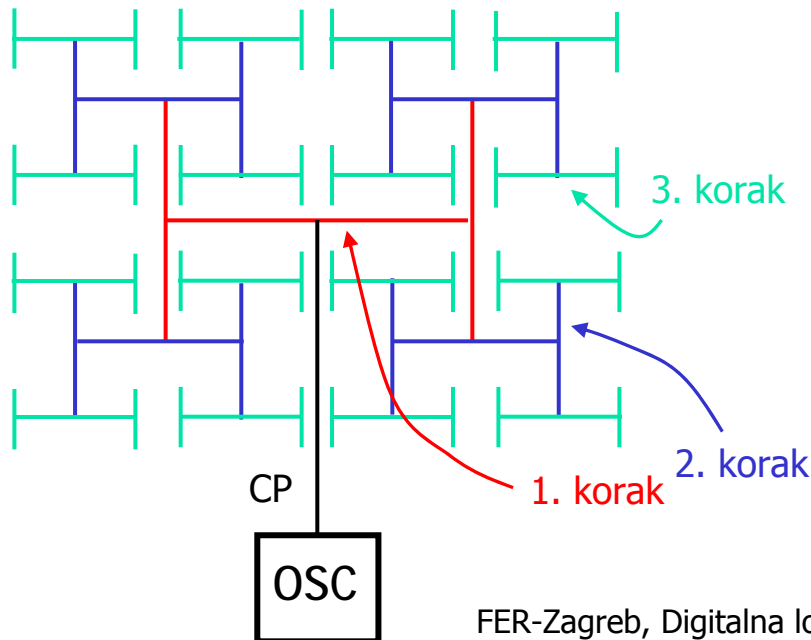
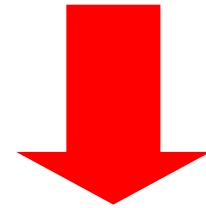
$t_{CS}$ : raskorak (engl. *clock skew time*)

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



# Raskorak

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



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



# 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