

Univerza v Ljubljani  
Fakulteta za elektrotehniko

# Digitalne strukture

Delovni zvezek za laboratorijske vaje

doc. dr. Gorazd Pucihar

Ime in priimek študenta:

---



## Navodila za laboratorijske vaje

### Splošno

Vaje potekajo v seminarju Laboratorija za biokibernetiko (stavba A, 3 nadstropje). Začnejo se 15 minut čez uro in trajajo 2 šolski uri (90 min), brez odmora. Vaje opravljate v parih, v skupinah, po razporedu in terminih, objavljenih na vratih laboratorija. Če vam predvideni termin vaj ne ustreza, lahko skupino zamenjate, če iz druge skupine najdete študenta, ki bo zamenjal z vami. Na koncu semestra bo razpisan dodaten termin za vse, ki ste manjkali na eni izmed vaj.

### Priprava na vajo

Pred vsako laboratorijsko vajo je potrebno rešiti naloge, ki so nujne za izdelavo vaje, njihov namen pa je priprava študentov na vajo. Rešene naloge prinesete na vajo, kjer jih asistent pregleda. Če naloge ne boste znali rešiti, se obrnite na asistenta vsaj en dan pred vajo. Naloge so sestavni del poročila, ki ga bo potrebno izdelati po vsaki vaji.

### Poročilo

Po vsaki uspešno ali neuspešno opravljeni laboratorijski vaji boste izdelali poročilo, ki ga prinesete na naslednjo vajo, skupaj s pripravo za tekočo vajo. Poročilo naj vsebuje:

- Naslov predmeta, ime in priimek študenta, šolsko leto in datum vaje.
- Številko vaje in besedilo vaje.
- Rešitev vaje (ponavadi je to priprava na vajo) + dodatne enačbe, tabele, diagrami
- Simbolni načrt vezja
- Vezalni načrt vezja (narisano na roko ali z ustreznim programskim orodjem)
- Spisek elementov + kratek komentar

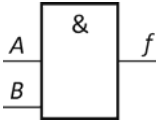

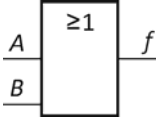
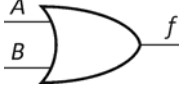
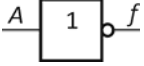

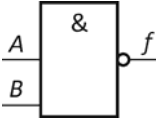

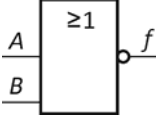

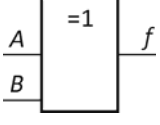

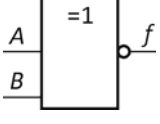

Poročila naj bodo kratka in jasna, s kratkimi komentarji, kjer je to potrebno. Poročilo izdelajte tudi, če vaja ni delovala, pri tem pa podajte možne razloge za nedelovanje.

### Ocenjevanje

Po vsaki laboratorijski vaji boste dobili tri ocene, sestavljene iz simbolov ("+", "o", "-", "0"), in sicer za pripravo na vajo, poročilo, ter samo vajo. Nedelujoča vaja se oceni z "-". Enako oceno dobite, če ne prinesete priprave ali poročila za prejšnjo vajo. Priprave ni možno prinesiti naknadno, medtem ko poročilo lahko prinesete naslednjič, vendar je v tem primeru najvišja ocena "o". Odsotnost pri vaji se oceni z "0". Končna ocena laboratorijskih vaj se izračuna iz povprečja ocen vseh vaj in je izražena v točkah (max. 100 točk). Ta se potem upošteva pri končni oceni predmeta. Obvezna je prisotnost na vsaj petih vajah. **Pozitivna ocena laboratorijskih vaj je pogoj za opravljanje pisnega dela izpita.**



## Osnovna logična vrata

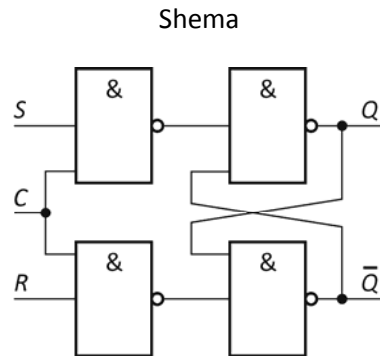
Ime	Funkcija	Simboli IEC	Simboli MIL
AND (IN)	$f = A \cdot B$		
OR (ALI)	$f = A + B$		
NOT (NE)	$f = \bar{A}$		
NAND (NEIN)	$f = \overline{A \cdot B}$		
NOR (NEALI)	$f = \overline{A + B}$		
XOR (IZKLJUČNO ALI)	$f = A \oplus B = \bar{A}B + A\bar{B}$		
NXOR (EKVIVALENCA)	$f = A \equiv B = \bar{A}\bar{B} + AB$		

A	B	AND	OR	NAND	NOR	XOR	NXOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1



# Pregled spominskih celic

## Zapah SR



Pravilnostna tabela

C	S	R	Q	$\bar{Q}$
0	X	X	Q	$\bar{Q}$
1	0	0	Q	$\bar{Q}$
1	0	1	0	1
1	1	0	1	0
1	1	1	1*	1*

X – poljubno, 0 ali 1

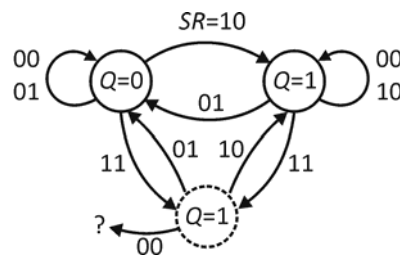
Karakteristična tabela  
( $Q_{(n)}$ : sedanje stanje,  $Q_{(n+1)}$ : naslednje stanje)

$S_{(n)}$	$R_{(n)}$	$Q_{(n)}$	$Q_{(n+1)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1*
1	1	1	1*

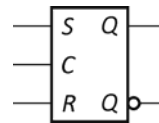
Vzbujalna tabela

$Q_{(n)}$	$Q_{(n+1)}$	$S_{(n)}$	$R_{(n)}$
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Diagram stanj



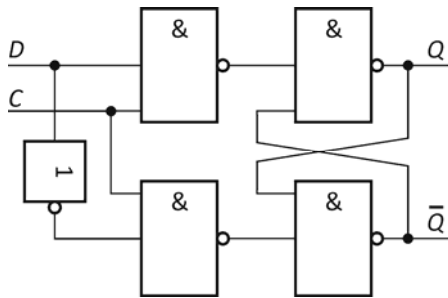
Simbol



X – poljubno, 0 ali 1

## Zapah D

Shema



Pravilnostna tabela

$C$	$D$	$Q$	$\bar{Q}$
0	X	$Q$	$\bar{Q}$
1	0	0	1
1	1	1	0

X – poljubno, 0 ali 1

Karakteristična tabela

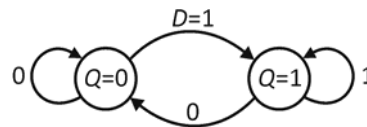
( $Q_{(n)}$ : sedanje stanje,  $Q_{(n+1)}$ : naslednje stanje)

$D_{(n)}$	$Q_{(n)}$	$Q_{(n+1)}$
0	0	0
0	1	0
1	0	1
1	1	1

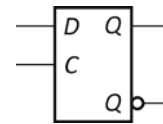
Vzbujalna tabela

$Q_{(n)}$	$Q_{(n+1)}$	$D_{(n)}$
0	0	0
0	1	1
1	0	0
1	1	1

Diagram stanj

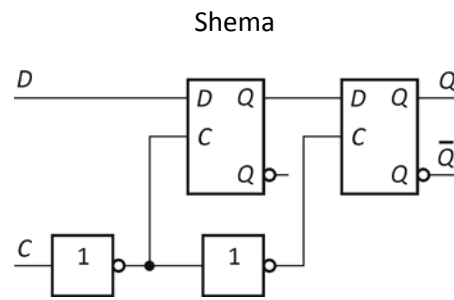


Simbol





## Flip-flop D



Pravilnostna tabela

C	D	Q	$\bar{Q}$
X	X	Q	$\bar{Q}$
$\downarrow$	0	0	1
$\uparrow$	1	1	0

X – poljubno, 0 ali 1

Karakteristična tabela

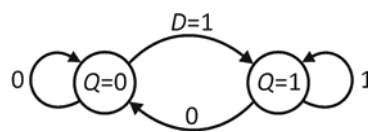
( $Q_{(n)}$ : sedanje stanje,  $Q_{(n+1)}$ : naslednje stanje)

$D_{(n)}$	$Q_{(n)}$	$Q_{(n+1)}$
0	0	0
0	1	0
1	0	1
1	1	1

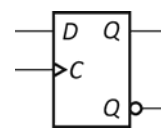
Vzbujalna tabela

$Q_{(n)}$	$Q_{(n+1)}$	$D_{(n)}$
0	0	0
0	1	1
1	0	0
1	1	1

Diagram stanj

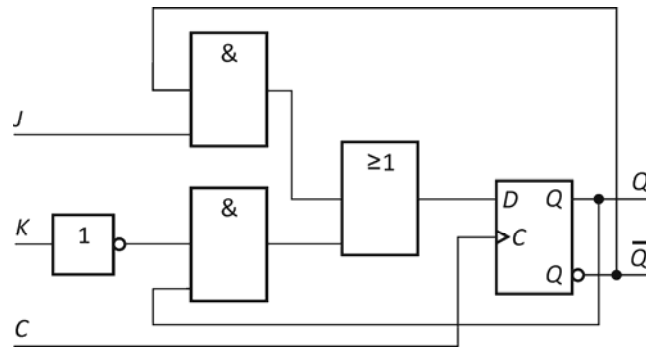


Simbol



# Flip-flop JK

Shema



Pravilnostna tabela

C	J	K	Q	$\bar{Q}$
X	X	X	Q	$\bar{Q}$
1	0	0	Q	$\bar{Q}$
1	0	1	0	1
1	1	0	1	0
1	1	1	$\bar{Q}$	Q

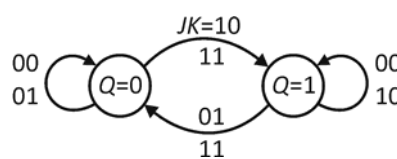
Karakteristična tabela  
( $Q_{(n)}$ : sed. stanje,  $Q_{(n+1)}$ : nasl. stanje)

$J_{(n)}$	$K_{(n)}$	$Q_{(n)}$	$Q_{(n+1)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

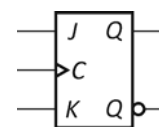
Vzbujalna tabela

$Q_{(n)}$	$Q_{(n+1)}$	$J_{(n)}$	$K_{(n)}$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Diagram stanj



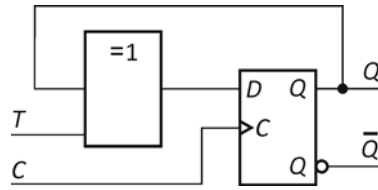
Simbol



X – poljubno, 0 ali 1

# Flip-flop T

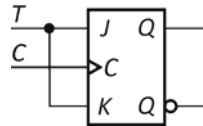
Shema (izvedba s f-f D)



Pravilnostna tabela

C	T	Q	$\bar{Q}$
X	X	Q	$\bar{Q}$
$\downarrow$	0	Q	$\bar{Q}$
$\downarrow$	1	$\bar{Q}$	Q

Shema (izvedba s f-f JK)



X – poljubno, 0 ali 1

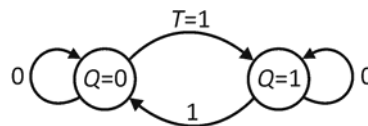
Karakteristična tabela  
( $Q_{(n)}$ : sedanje stanje,  $Q_{(n+1)}$ : naslednje stanje)

$T_{(n)}$	$Q_{(n)}$	$Q_{(n+1)}$
0	0	0
0	1	1
1	0	1
1	1	0

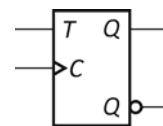
Vzbujalna tabela

$Q_{(n)}$	$Q_{(n+1)}$	$T_{(n)}$
0	0	0
0	1	1
1	0	1
1	1	0

Diagram stanj



Simbol





## Primer poročila za laboratorijsko vajo

**Naloga:** Funkcijo, ki ste jo dobili na listu, zapišite v popolni in minimalni disjunktivni in konjunktivni normalni obliki (PDNO, PKNO, MDNO, MKNO). Funkcijo v minimalni obliki nato na protoboardu realizirajte z logičnimi vrati NAND.

**Primer funkcije, ki jo moramo realizirati z vrati NAND:**

$$f(x_1, x_2, x_3) = \sum 2, 3, 4, 6$$

**Rešitev:**\*

$$f_{PDNO} = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3$$

$$f_{MDNO} = x_1 \bar{x}_3 + \bar{x}_1 x_2$$

$$\begin{array}{cccccccc}
 m_0 & m_1 & \cancel{m_2} & \cancel{m_3} & \cancel{m_4} & m_5 & \cancel{m_6} & m_7 \\
 \downarrow & \downarrow & & & & \downarrow & & \downarrow \\
 M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0
 \end{array}$$

$$f_{PKNO} = (\bar{x}_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)(x_1 + x_2 + \bar{x}_3)(x_1 + x_2 + x_3)$$

$$f_{MKNO} = (x_1 + x_2)(\bar{x}_1 + \bar{x}_3)$$

$x_1$	$x_2$	$x_3$	$f$	
0	0	0	0	
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	0	

Realizacija z NAND: funkcijo v MDNO 2x negiramo:

$$f_{MDNO} = \overline{\overline{f_{MDNO}}} = \overline{x_1 \bar{x}_3 + \bar{x}_1 x_2} = \overline{(x_1 \bar{x}_3) \cdot (\bar{x}_1 x_2)}$$

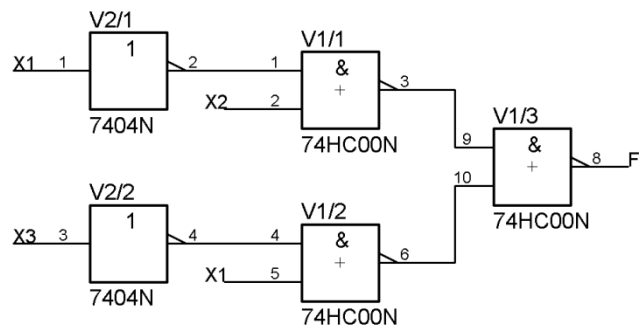
		$x_1 x_2$			
		00	01	11	10
$x_3$	0		1	1	1
	1		1		

		$x_1 x_2$			
		00	01	11	10
$x_3$	0	0			
	1	0		0	0

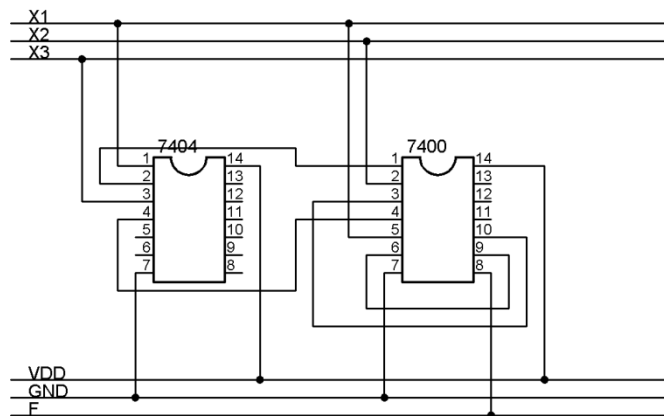
\* Izpolnite kot pripravo na vajo.

**Simbolni načrt vezja:**

(npr. s programskim orodjem Eagle)

**Vezalni načrt vezja:**

(npr. s programskim orodjem Eagle)

**Spisek uporabljenih elementov:****1×74HC00****1×74HC04****Komentar:**

Pri vaji nismo imeli težav...

Vezje ni delovalo, ker ....

Za realizacijo vaje nam je zmanjkalo časa...

Ugotovili smo, ....

- .
- .
- .

\* Izpolnite kot pripravo na vajo.

## VAJA 1: Realizacija primerjalnika z logičnimi vrati

Ime in priimek:

Datum:

Sodelavec:

Skupina:

**Naloga 1a:** Zgradite primerjalnik dveh enobitnih števil  $A$  in  $B$ , kot zahteva naloga na listu. Vezje zgradite na protoboardu z uporabo logičnih vrat AND in NOT.

**Rešitev (tabela, enačba):\***

$A$	$B$	$f$

**Simbolni načrt vezja :**

**Vežalni načrt vezja:**

---

\* Izpolnite kot pripravo na vajo.

**Naloga 1b:** Zgradite primerjalnik dveh dvobitnih števil  $A (A_1A_0)$  in  $B (B_1B_0)$ , kot zahteva naloga na listu. Izhodno funkcijo zapišite v popolni in minimalni disjunktivni in konjunktivni normalni obliki (PDNO, PKNO, MDNO, MKNO). Funkcijo v minimalni obliki nato realizirajte na protoboardu z logičnimi vrati NAND oz. NOR.

**Rešitev:\***

$A_1$	$A_0$	$B_1$	$B_0$	$f$	

<table border="1" style="margin: auto;"> <tr><td colspan="4"></td></tr> <tr><td colspan="4" style="text-align: center;"><math>A_1A_0</math></td></tr> <tr><td></td><td style="text-align: center;">00</td><td style="text-align: center;">01</td><td style="text-align: center;">11</td><td style="text-align: center;">10</td></tr> <tr><td style="text-align: center;">00</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">01</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">11</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">10</td><td></td><td></td><td></td><td></td></tr> </table>					$A_1A_0$					00	01	11	10	00					01					11					10					<table border="1" style="margin: auto;"> <tr><td colspan="4"></td></tr> <tr><td colspan="4" style="text-align: center;"><math>A_1A_0</math></td></tr> <tr><td></td><td style="text-align: center;">00</td><td style="text-align: center;">01</td><td style="text-align: center;">11</td><td style="text-align: center;">10</td></tr> <tr><td style="text-align: center;">00</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">01</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">11</td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">10</td><td></td><td></td><td></td><td></td></tr> </table>					$A_1A_0$					00	01	11	10	00					01					11					10				
$A_1A_0$																																																																			
	00	01	11	10																																																															
00																																																																			
01																																																																			
11																																																																			
10																																																																			
$A_1A_0$																																																																			
	00	01	11	10																																																															
00																																																																			
01																																																																			
11																																																																			
10																																																																			

\* Izpolnite kot pripravo na vajo.



**Simbolni načrt vezja:\***

**Vežalni načrt vezja:**

**Spisek  
uporabljenih  
elementov:**

**Komentar:**

---

\* Izpolnite kot pripravo na vajo.

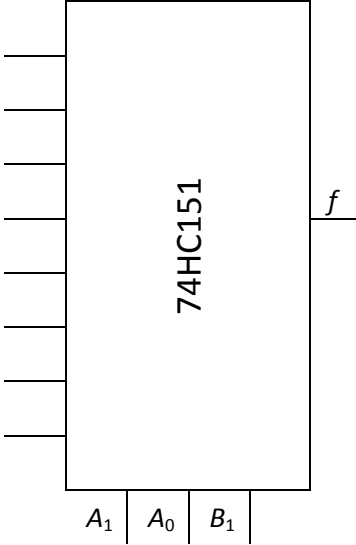


## VAJA 2: Kombinacijska vezja: multipleksor, dekodirnik

Ime in priimek:		Datum:
Sodelavec:		Skupina:

**Naloga 2a:** Dvobitni primerjalnik iz *Vaje 1b* zgradite z multipleksorjem 74HC151, ki ima 3 naslovne in 8 podatkovnih vhodov. Na naslovne vhode priključite vhodne spremenljivke  $A_1$ ,  $A_0$  in  $B_1$ . Na voljo imate še vrata NOT.

**Rešitev:**\*



**Vežalni načrt primerjalnika z multipleksorjem 74HC151:**

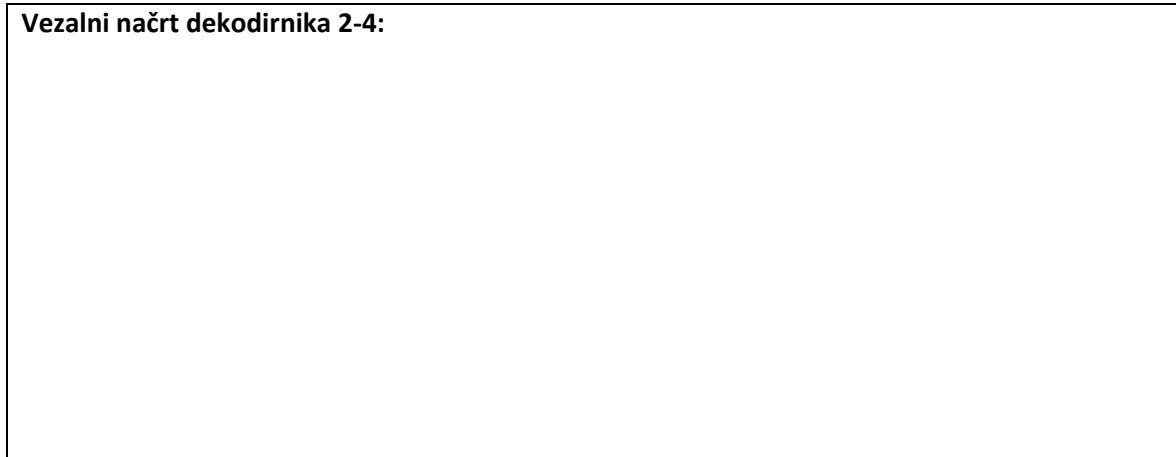
---

\* Izpolnite kot pripravo na vajo.

**Naloga 2b:** Z logičnim vezjem 74HC139 zgradite dekodirnik 2-4, ki ima dva naslovna vhoda  $A_1$  in  $A_0$  in vhod enable, ter štiri izhode  $f$ . Delovanje dekodirnika prikazuje tabela:

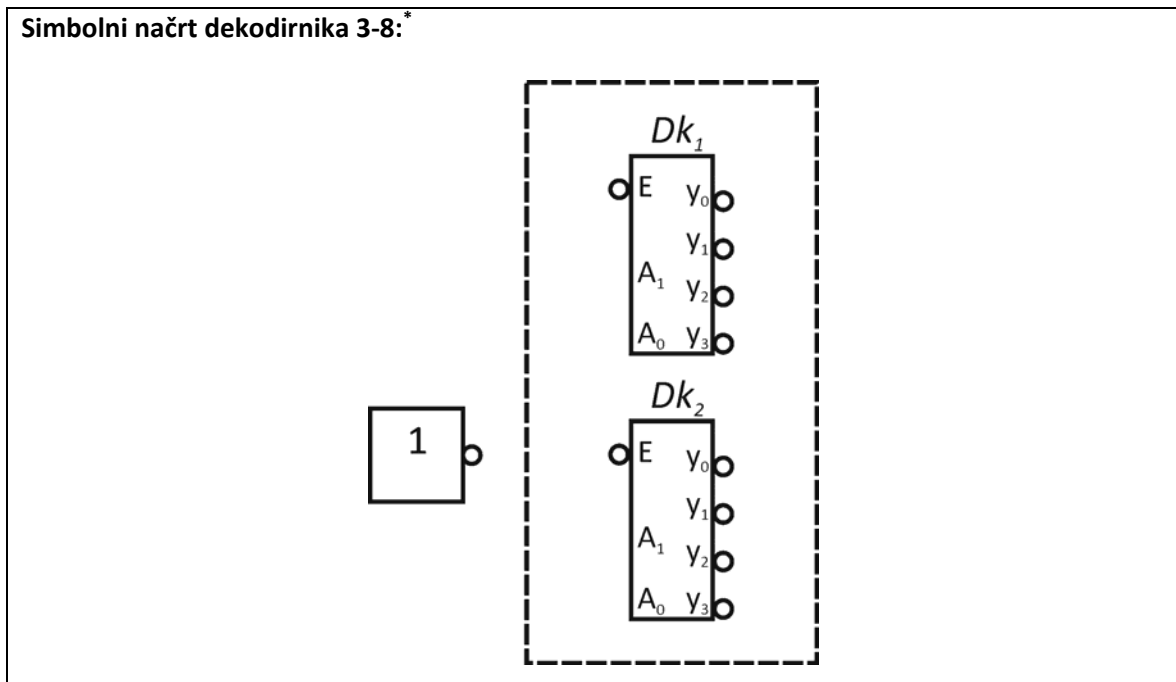
$E$	$A_1$	$A_0$	$f_3$	$f_2$	$f_1$	$f_0$
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

**Vežalni načrt dekodirnika 2-4:**



**Naloga 2c:** Z dvema dekodirnikoma 2-4 (74HC139) zgradite dekodirnik 3-8. Na voljo imate še vrata NOT. Dekodirnik naj ima tri naslovne vhode  $A_2$ ,  $A_1$ ,  $A_0$  in vhod enable, ter osem izhodov  $f_i$ . Tabela delovanja je podobna tabeli za dekodirnik 2-4.

**Simbolni načrt dekodirnika 3-8:**\*



\* Izpolnite kot pripravo na vajo.

**Vežalni načrt dekodirnika 3-8:**

**Spisek  
uporabljenih  
elementov:**

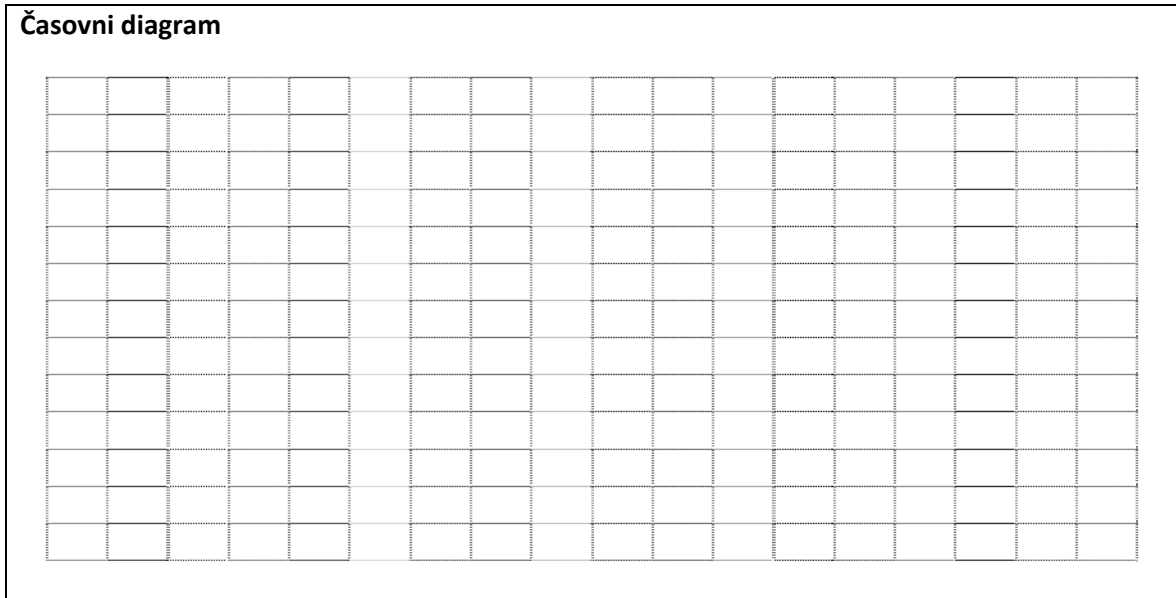
**Komentar:**



**VAJA 3: Simulacija delovanja kombinacijskih vezij s programskim orodjem P-SPICE**

<b>Ime in priimek:</b>		<b>Datum:</b>
<b>Sodelavec:</b>		<b>Skupina:</b>

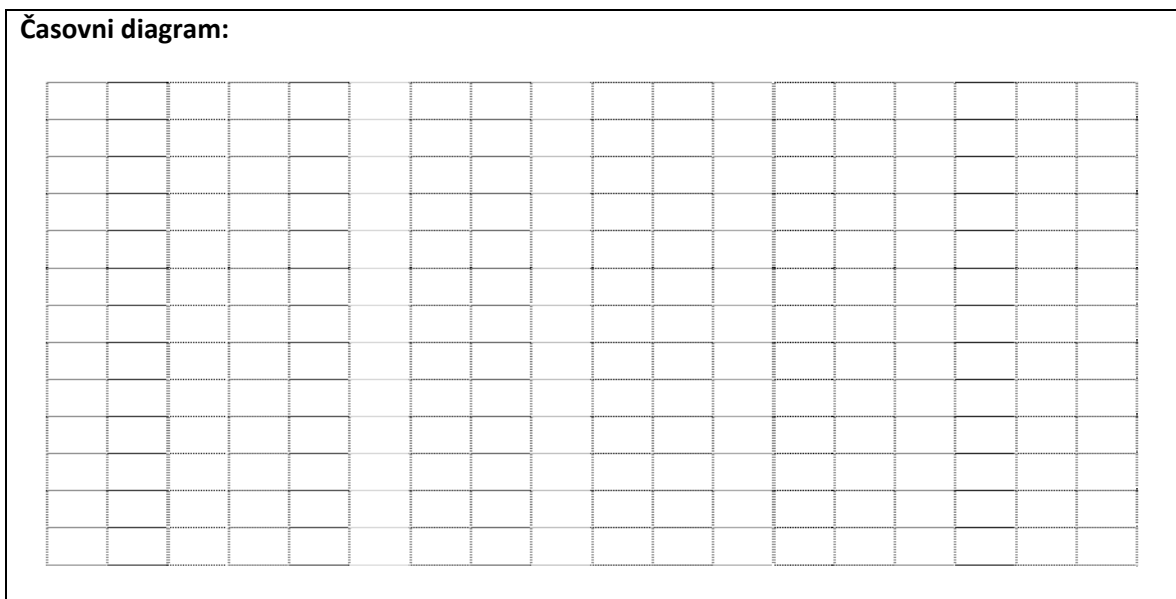
**Naloga 3a:** Delovanje dvobitnega primerjalnika iz *Vaje 1, Naloge 1b* simulirajte v programskem okolju P-SPICE. Skicirajte časovne diagrame vhodnih in izhodne funkcije.



**Naloga 3b:** Določite zakasnitev  $\Delta t$  na vratih NAND oz. NOR.

$\Delta t_{\text{NAND/NOR}}$ : \_\_\_\_\_

**Naloga 3c:** Simulirajte delovanje multipleksorja iz *Vaje 2, Naloge 2a* in narišite časovni diagram vhodnih in izhodne funkcije.



**Naloga 3d.** Funkcijo, podano na listu, zapišite v minimalni disjunktivni oz. konjunktivni obliki (AND-OR oz. OR-AND). Simulirajte delovanje vezja in ugotovite pri kateri spremembi vhodnih spremenljivk se v vezju pojavi statični hazard.

**Funkcija:**

**Rešitev:\***

		$x_1x_2$			
		00	01	11	10
$x_3$	0				
	1				

**Simbolni načrt:\***

---

\* Izpolnite kot pripravo na vajo.



**Časovni diagram vezja s hazardom:**


Hazard se pojavi pri naslednji spremembi vhodnih spremenljivk: \_\_\_\_\_

**Naloga 3e:** Vezje iz *Naloga 3d* dopolnite tako, da boste hazard odpravili.

**Simbolni načrt vezja brez hazarda:**\*

---

\* Izpolnite kot pripravo na vajo.



## VAJA 4: Opis digitalnih vezij z jezikom VHDL

Ime in priimek:		Datum:
Sodelavec:		Skupina:

**Naloga 4a:** S strojno opisnim jezikom VHDL zapišite delovanje enobitnega primerjalnika iz *Vaje 1, Naloge 1a*.

### VHDL koda enobitnega primerjalnika:\*

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

entity primerjalnik1a is
  Port (A: in STD_LOGIC;
        B: in STD_LOGIC;
        f: out STD_LOGIC);
end primerjalnik1a;

architecture Behavioral of primerjalnik1a is

begin

end Behavioral;
```

**Naloga 4b:** S strojno opisnim jezikom VHDL zapišite delovanje dvobitnega primerjalnika iz *Vaje 1, Naloge 1b*. Izhajajte iz zapisa MDNO, posamezne produkte v funkciji pa zapišite kot nove spremenljivke (npr.  $S_1 = A_0A_1B_1$ ).

### VHDL koda dvobitnega primerjalnika:\*

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

entity primerjalnik1b is
  Port (A: in STD_LOGIC_VECTOR (1 downto 0);
        B: in STD_LOGIC_VECTOR (1 downto 0);
        f: out STD_LOGIC);
end primerjalnik1b;

architecture Behavioral of primerjalnik1b is
```

---

\* Izpolnite kot pripravo na vajo.

```
begin
```

```
end Behavioral;
```

**Naloga 4c:** S strojno opisnim jezikom VHDL zapišite delovanje multipleksorja iz *Vaje 2, Naloga 2a* (uporabite stavke *when...else*).

**VHDL koda multipleksorja:**\*

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

entity mux is
  Port (A: in STD_LOGIC_VECTOR (1 downto 0);
        B: in STD_LOGIC_VECTOR (1 downto 0);
        f: out STD_LOGIC);
end mux;

architecture Behavioral of mux is

begin

end Behavioral
```

---

\* Izpolnite kot pripravo na vajo.

**Naloga 4d:** S strojno opisnim jezikom VHDL zapišite delovanje dekodirnika 2-4 iz *Vaje 2, Naloge 2b* (uporabite stavke *with... select... when*).

**VHDL koda dekodirnika:**\*

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

entity dekodirnik is
  Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
        Enable : in STD_LOGIC;
        Y : out STD_LOGIC_VECTOR (3 downto 0));
end dekodirnik;

architecture Behavioral of dekodirnik is

begin

end Behavioral;
```

---

\* Izpolnite kot pripravo na vajo.



## Pomoč pri pripravi na 4. laboratorijsko vajo (VHDL)

- Definicija vmesnika (stavek *entity*):

```
entity ime_vezja is
  port (definicija vhodov/izhodov);
end ime_vezja;
```

- Definicija vhodov/izhodov (stavek *port*):

```
port ( A,B,C,... : in STD_LOGIC;
      d,e,f,... : in STD_LOGIC_VECTOR (3 downto 0);      -- štiribitni vektorji d, e,f
      f1,f2,f3,... : out STD_LOGIC);
```

- Definicija opisa vezja (stavek *architecture*):

```
architecture opis of ime_vezja is
:
definicije spremenljivk; -- tu določimo vse spremenljivke, ki nastopajo v opisu vezja in niso
:                          --določene v stavku port
Begin                      -- stavku architecture (za definicijo spremenljivk) vedno sledi stavek
                          --begin
:
stavki;                    -- vse stavke/ukaze pišemo znotraj stavka begin
:
end opis;
```

- Definicija spremenljivk (stavek *signal*):

-Enobitne spremenljivke: -- vedno znotraj stavka *architecture*, pred stavkom *begin*

```
signal A,B,x, y, z: std_logic;
```

-Večbitne spremenljivke/vektorji:

```
signal w, q, r : std_logic_vector (3 downto 0); -- štiribitni vektorji w, q, r
```

- Prirejanje vrednosti spremenljivkam:

-Enobitne spremenljivke:

```
A<='1'; -- spremenljivki A se priredi vrednost 1
```

```
A='1'; -- vrednost spremenljivke A se primerja z '1'. Rezultat je true/false.
```

*Primer:*

```
f<='1' when A='1' else -- spremenljivka f zavzame vrednost 1, če je spremenljivka A enaka 1,
'0';                -- sicer je f =0.
```

-Večbitne spremenljivke/vektorji:

*Primer:*

```
B<="1011"; -- vektorju B se priredi vrednost 1011
```

```
B(3)<='0'; -- bitu z indeksom 3 vektorja B ( $B=B_3B_2B_1B_0$ ) se priredi vrednost 0,  $B=0011$ 
```

```
B<=B+1; -- B se poveča za vrednost 1,  $B=4_{(10)}=0100_{(2)}$ 
```

- Združevanje signalov (operator "&")  
Primer:  
A<='1';  
B<='0';  
C<="110";  
f<=A&B&C;            -- vektorju f se priredi vrednost f=10110
- Stavek *when...else*:  
ime\_spremenljivke<=vrednost1 when izraz1 else  
vrednost2 when izraz2 else  
...  
vrednost n;                            -- na koncu stavka je podpičje
- Stavek *with...select...when*:  
with ime\_izbirnega\_signala select  
ime\_spremenljivke <=vrednost 1 when izbor 1, -- na koncu vsakega izbora je vejica  
vrednost 2 when izbor 2,  
...  
vrednost n when others;            -- na koncu stavka je podpičje
- Stavek *process*:                    -- opis vezja (*architecture*) lahko vsebuje več stavkov *process*  
*process* (spisek spremenljivk, ki nastopijo znotraj tega stavka)  
begin                                --stavek *process* ima svoja stavka *begin* in *end*  
:  
sekvenčni stavki;            --stavek *process* se uporablja za zapis sekvenčnih/zaporednih ukazov  
:  
end process;
- Stavek *if...then...else*:            --stavek *if* je sekvenčni stavek in se vedno uporablja znotraj stavka *process*  
if izraz 1 then  
stavki;  
elsif izraz 2 then  
stavki;  
:  
else izraz n;  
end if;                                --stavek *if* ima svoj stavek *end*
- Proženje na naraščajoči del CLK pulza (prednja fronta):  
if (clk'event and clk='1') then  
stavki;  
end if;

VHDL ne loči med spremenljivkami zapisanimi z veliko in malo začetnico (A=a).

Vsak prireditveni stavek (npr. f<= a and b) se mora zaključiti s podpičjem. Podpičje je tudi za vsakim stavkom *end*.



## VAJA 5: Sekvenčna vezja: Spominske celice

Ime in priimek:

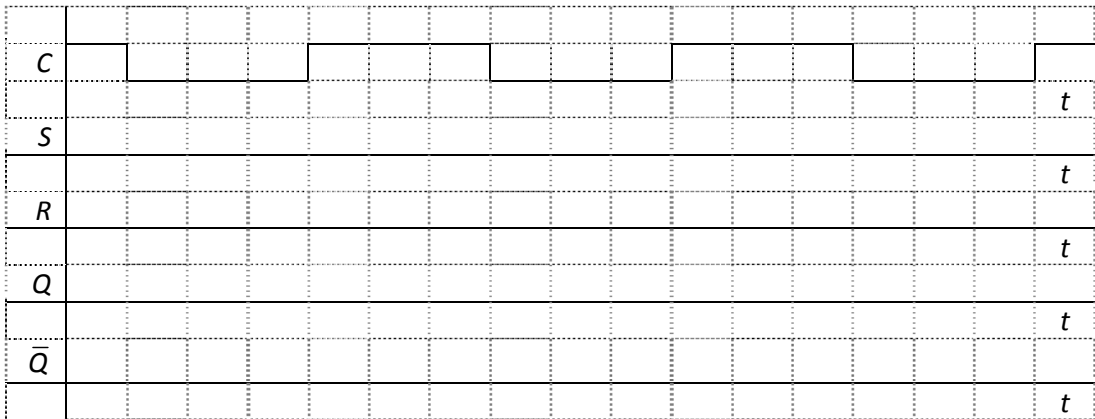
Datum:

Sodelavec:

Skupina:

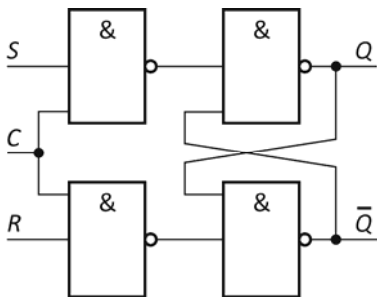
**Naloga 5a:** Narišite časovni diagram izhodov  $Q$  in  $\bar{Q}$  zapaha SR, če je podan časovni diagram vhodnih spremenljivk (glej diagram na listu).

Rešitev:\*



**Naloga 5b:** Z logičnimi vrati NAND na protoboardu realizirajte vezje zapaha SR. Vhodne spremenljivke spreminjajte tako, kot prikazuje časovni diagram v *Nalogi 5a* in preverite, če na izhodih  $Q$  in  $\bar{Q}$  dobite predvidene vrednosti. Nato vhodni spremenljivki  $S$  in  $R$  sočasno večkrat spremenite iz 11 v 00 in vsakič zapišite vrednosti izhodov. Podajte razlago za obnašanje vezja pri teh prehodih.

Simbolni načrt zapaha SR:

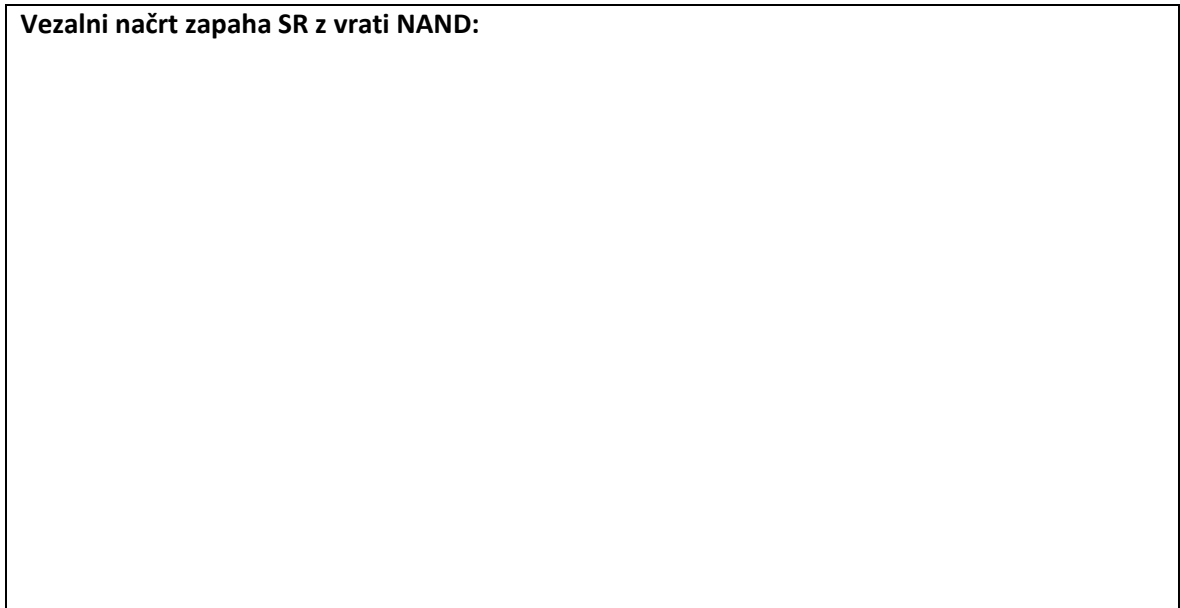


$C$	$S$	$R$	$Q$	$\bar{Q}$
1	1	1		
1	0	0		
1	1	1		
1	0	0		
1	1	1		
1	0	0		
1	1	1		
1	0	0		
1	1	1		
1	0	0		

Razlaga obnašanja vezja pri prehodu  $SR$  iz 11 v 00:

\* Izpolnite kot pripravo na vajo.

**Vežalni načrt zapaha SR z vrati NAND:**



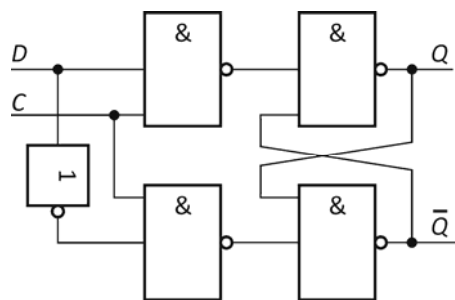
**Naloga 5c:** Narišite časovni diagram izhoda  $Q$  zapaha  $D$ , če je podan časovni diagram vhodnih spremenljivk (glej diagram na listu).

**Rešitev:**\*

$C$																		$t$
$D$																		$t$
$Q$																		$t$

**Naloga 5d:** Z logičnimi vrati NAND in NOT na protoboardu realizirajte vezje zapaha  $D$ . Vhodne spremenljivke spreminjajte tako, kot prikazuje časovni diagram v *Nalogi 5c* in preverite, če na izhodu  $Q$  dobite predvidene vrednosti.

**Simbolni načrt zapaha D:**

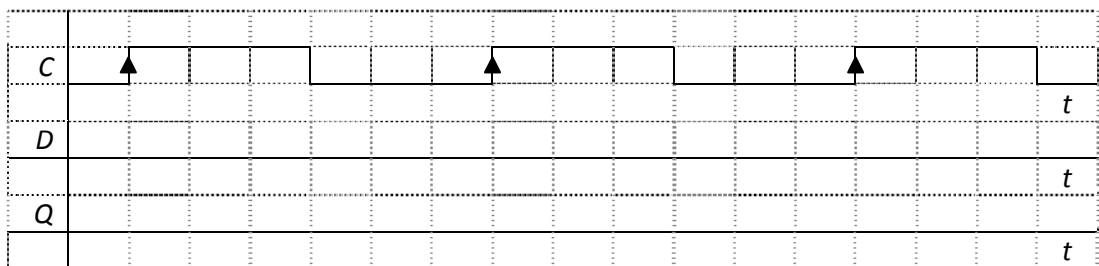


\* Izpolnite kot pripravo na vajo.

**Vežalni načrt zapaha D z vrati NAND in NOT:**

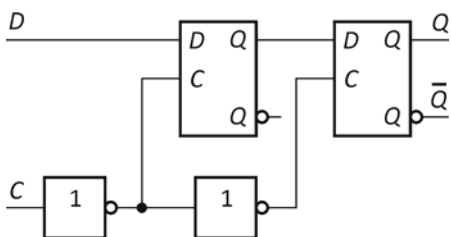
**Naloga 5e:** Narišite časovni diagram izhoda  $Q$  flip-flopa D, če je podan časovni diagram vhodnih spremenljivk (glej diagram na [listu](#)).

**Rešitev:**\*



**Naloga 5f:** Z logičnimi vrati NAND in NOT na protoboardu realizirajte vezje flip-flopa D. Vhodne spremenljivke spreminjajte v zaporedju, prikazanem v časovnem diagramu *Naloga 5e* in preverite, če na izhodu  $Q$  dobite predvidene vrednosti.

**Simbolni načrt flip-flopa D:**



**Pri katerem prehodu spremenljivke  $C$  se spremeni izhod  $Q$ ?**

\* Izpolnite kot pripravo na vajo.

**Vežalni načrt flip-flopa D:**

**Spisek  
uporabljenih  
elementov:**

**Komentar:**

## VAJA 6: Sekvenčna vezja: sinhronski števec

Ime in priimek:		Datum:
Sodelavec:		Skupina:

**Naloga 6a:** Zgradite sinhronski števec, ki šteje gor/dol tako, kot določajo navodila na listu. Narišite diagram prehajanja stanj, zapišite tabelo prehajanja stanj in vzbujalno tabelo ter podajte simbolni načrt vezja. Vezje nato realizirajte na protoboardu.

Modul štetja: \_\_\_\_\_

Smer štetja: \_\_\_\_\_

Komb. del: \_\_\_\_\_

Spom. del: \_\_\_\_\_

Diagram prehajanja stanj:\*

Tabela stanj in kodirana tabela stanj:\*

---

\* Izpolnite kot pripravo na vajo.

**Vzbujalna tabela:\***

**Simbolni načrt vezja:\***

---

\* Izpolnite kot pripravo na vajo.

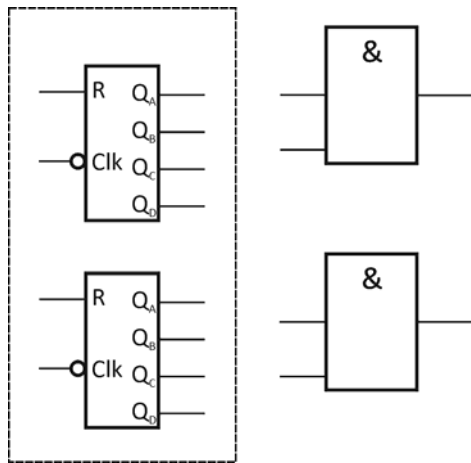
**Vežalni načrt vezja:**

**Naloga 6b:** Z vezjem SN74LS390 zgradite desetiški sinhronski števec (števec, ki šteje od 0 do 9).

**Vežalni načrt desetiškega števca z SN74LS390:**

**Naloga 6c:** Števec iz *Naloga 6b* dopolnite tako, da bo štel od 00 do 59. Uporabite vezje SN74LS390 in logična vrata AND.

**Simbolni načrt števca 00-59:**\*



**Vežalni načrt števca 00-59:**

---

\* Izpolnite kot pripravo na vajo.



**Spisek  
uporabljenih  
elementov:**

**Komentar:**

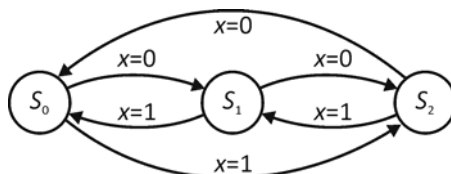


## Pomoč pri pripravi na 6. laboratorijsko vajo

**Naloga:** Zgradite sinhronski števec, ki šteje navzgor in navzdol po modulu 3. Smer štetja naj določa vhodna spremenljivka  $x$  ( $x = 0$ , štetje navzgor;  $x = 1$ , štetje navzdol). Spominski del vezja realizirajte s celicami JK, kombinacijskega pa z minimalnim številom logičnih vrat.

**Opomba:** Nekateri morate zgraditi števec, ki šteje le navzgor. V tem primeru ne boste potrebovali vhodne spremenljivke  $x$ .

**Rešitev:** Modul štetja pove, koliko stanj ima vezje, v našem primeru torej 3 stanja. Števec naj šteje navzgor ...0, 1, 2, 0, 1... (ko bo  $x = 0$ ) in navzdol ...2, 1, 0, 2, 1... (ko bo  $x = 1$ ). Iz teh podatkov lahko narišemo *diagram stanj*:



Iz diagrama stanj zapišemo *tabelo stanj*. Nato stanja zakodiramo (če ni določeno, uporabimo kar naravno binarno kodo). Za kodiranje štirih stanj zadoščata dve spremenljivki stanj ( $Q_1$  in  $Q_2$ ), kar pomeni, da bomo za realizacijo vezja potrebovali dve spominski celici JK.

Tabela stanj

Sedanje stanje	Naslednje stanje	
	$x = 0$	$x = 1$
$S_0$	$S_1$	$S_2$
$S_1$	$S_2$	$S_0$
$S_2$	$S_0$	$S_1$

Kodirana tabela stanj

Stanje	$Q_1$	$Q_2$
$S_0$	0	0
$S_1$	0	1
$S_2$	1	0

S pomočjo tabele stanj in kodirane tabele stanj zapišemo *vzbujalno tabelo* ( $Q_{i(n)}$ -sedanje stanje,  $Q_{i(n+1)}$  – naslednje stanje, X – poljubna vrednost, lahko 0 ali 1):

Vzbujalna tabela

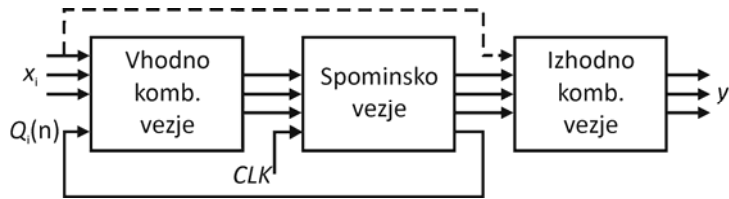
$x$	$Q_{1(n)}$	$Q_{2(n)}$	$Q_{1(n+1)}$	$Q_{2(n+1)}$	$J_1$	$K_1$	$J_2$	$K_2$
0	0	0	0	1	0	X	1	X
0	0	1	1	0	1	X	X	1
0	1	0	0	0	X	1	0	X
0	1	1	X	X	X	X	X	X
1	0	0	1	0	1	X	0	X
1	0	1	0	0	0	X	X	1
1	1	0	0	1	X	1	1	X
1	1	1	X	X	X	X	X	X

Stolpce za  $J_1$ ,  $K_1$  in  $J_2$ ,  $K_2$  dobimo s pomočjo vzbujalne tabele za celico JK (glej "Pregled spominskih celic"). Na primer,  $J_1$  in  $K_1$  v drugi vrstici vzbujalne tabele dobimo tako, da pogledamo kakšna morata biti  $J$  in  $K$ , da se bo spremenljivka stanj  $Q_1$  spremenila iz  $Q_{1(n)}$  v  $Q_{1(n+1)}$ , torej iz 0 v 1. Iz tabele za celico JK razberemo, da bomo želeno spremembo dobili, ko bosta:  $J_1 = 1$  in  $K_1 = X$ .

$Q_{1(n)}$	$Q_{1(n+1)}$	$J_1$	$K_1$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Sedaj je potrebno zgraditi še vezje (glej Sliko 1). *Spominski del vezja* bo sestavljen iz dveh spominskih celic JK. Da dobimo zahtevane spremembe spremenljivk stanj na izhodu celic,  $Q_{(n)} \rightarrow Q_{(n+1)}$ , potrebujemo še ustrezno *vhodno kombinacijsko vezje*, ki bo na vhode spominskih celic ( $J_1$ ,  $K_1$ ,  $J_2$ ,  $K_2$ )

pripeljalo ustrezne signale, tako kot to določa vzbujačna tabela. Na primer, če pogledamo peto vrstico vzbujačne tabele vidimo, da bomo pri  $x = 1$ , ob CLK pulzu, prešli iz stanja  $S_0$  ( $Q_{1(n)} Q_{2(n)} = 00$ ) v stanje  $S_2$  ( $Q_{1(n+1)} Q_{2(n+1)} = 10$ ). Ta sprememba stanja pa se bo zgodila le, če bodo vhodi v spominske celice (oz. izhodi iz kombinacijskega vezja)  $J_1 = 1$ ,  $K_1 = X$ ,  $J_2 = 0$  in  $K_2 = X$ . Vhodi v kombinacijsko vezje bodo torej  $x$ ,  $Q_{1n}$ ,  $Q_{2n}$ , izhodi iz kombinacijskega vezja pa  $J_1, K_1, J_2, K_2$ .



Slika1. Shema sekvenčnega vezja.

Naloga zahteva, da kombinacijsko vezje realiziramo z minimalnim številom logičnih vrat. Z uporabo K-diagramov dobimo minimalni zapis funkcij  $J_1, K_1, J_2$  in  $K_2$ , ki jih nato realiziramo z logičnimi vrati. Izhod sekvenčnega vezja je v našem primeru kar izhod iz celic JK ( $Q_1, Q_2$ ), v splošnem pa ima sekvenčno vezje lahko še izhodno kombinacijsko vezje.

$J_1$ :

		$xQ_{1(n)}$			
		00	01	11	10
$Q_{2(n)}$	0	0	X	X	1
	1	1	X	X	0

$K_1$ :

		$xQ_{1(n)}$			
		00	01	11	10
$Q_{2(n)}$	0	X	1	1	X
	1	X	X	X	X

$J_2$ :

		$xQ_{1(n)}$			
		00	01	11	10
$Q_{2(n)}$	0	1	0	1	0
	1	X	X	X	X

$K_2$ :

		$xQ_{1(n)}$			
		00	01	11	10
$Q_{2(n)}$	0	X	X	X	X
	1	1	X	X	1

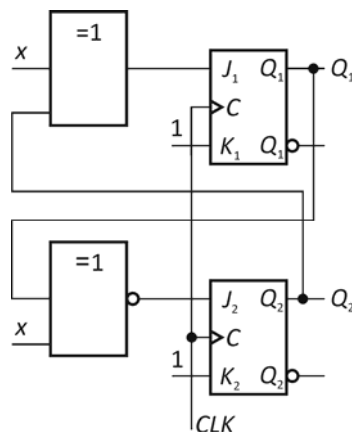
$$J_1(x, Q_{1(n)}, Q_{2(n)}) = x\bar{Q}_{2(n)} + \bar{x}Q_{2(n)} = x \oplus Q_{2(n)}$$

$$K_1(x, Q_{1(n)}, Q_{2(n)}) = 1$$

$$J_2(x, Q_{1(n)}, Q_{2(n)}) = xQ_{1(n)} + \bar{x}\bar{Q}_{1(n)} = x \equiv Q_{1(n)}$$

$$K_2(x, Q_{1(n)}, Q_{2(n)}) = 1$$

Vezje:



## VAJA 7: Realizacija sinhronskih sekvenčnih vezij s programirljivim vezjem FPGA

Ime in priimek:		Datum:
Sodelavec:		Skupina:

**Naloga 7a:** S strojno opisnim jezikom VHDL zapišite delovanje sinhronskega števec, kot določajo navodila na listu (uporabite stavka *process* in *if...then*). Glej tudi *Pomoč pri pripravi na 4. laboratorijsko vajo (VHDL)*.

VHDL koda števec:\*

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

entity stevec is
  Port (clk : in STD_LOGIC;
        q : out STD_LOGIC_VECTOR (1 downto 0));
end stevec;

architecture Behavioral of stevec is

begin
  process ( )
    begin

end process;

end Behavioral;
```

---

\* Izpolnite kot pripravo na vajo.

**Naloga 7b:** Zapišite VHDL kodo za vezje, ki bo frekvenco ure na plošči Xilinx zmanjšalo iz 50 MHz na 1 Hz (Namig: delilnik frekvence lahko naredimo s števcem).

**VHDL koda za delilnik frekvence ure:**\*

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

entity delilnik is
  Port (clk_in : in STD_LOGIC; -- CLK 50 MHz iz plošče Xilinx
        clk : out STD_LOGIC; -- CLK 1 Hz
        );
end delilnik;

architecture Behavioral of delilnik is

begin
  process (clk_in)
  begin

end process;

end Behavioral;
```

---

\* Izpolnite kot pripravo na vajo.

**Naloga 7c:** Zgradite sinhronsko sekvenčno vezje v Mooreovi obliki, ki bo vklapljalno smernike (LED) pri avtomobilu. Pri tem naj vhodni spremenljivki  $L$  in  $D$  določata, kateri od obeh smernikov bo utripal (utripanje je potrebno zagotoviti s samim vezjem). Vhodna spremenljivka  $E$ , ki ima največjo težo, naj omogoči vklop varnostnih smernikov, ko bosta utripala oba smernika hkrati. Narišite diagram in tabelo stanj, vezje pa realizirajte z uporabo programirljivega vezja FPGA.

**Diagram stanj (le osnovni prehodi):\***

**Tabela stanj:\***

Sed. stanje	Naslednje stanje								Izhodi	
	$LDE$	$LDE$	$LDE$	$LDE$	$LDE$	$LDE$	$LDE$	$LDE$	Leva	Desna
	000	001	010	011	100	101	110	111		
$S_0$										
$S_1$										
$S_2$										
$S_3$										

\* Izpolnite kot pripravo na vajo.

**Naloga 7d:** Zgradite krmilno vezje za semafor v Mooreovi obliki, ki naj deluje po naslednjem pravilu:

- V dnevnem načinu delovanja ( $D = 1$ ) se odvija sekvenca: zelena – rumena – rdeča – rumena – zelena... Pri tem naj zelena ( $y_{Ze}$ ) in rdeča luč ( $y_{Rd}$ ) svetita tri urine periode, rumena ( $y_{Ru}$ ) pa eno urino periodo.
- V nočnem načinu delovanja ( $D = 0$ ), oziroma v primeru napake, naj utripa le rumena luč.

Narišite diagram in tabelo stanj, vezje pa realizirajte z uporabo programirljivega vezja FPGA.

**Diagram stanj:**\*

**Tabela stanj:**\*

Sed. stanje	Nasl. stanje		Izhodi		
	$D = 0$	$D = 1$	$y_{Rd}$	$y_{Ru}$	$y_{Ze}$

---

\* Izpolnite kot pripravo na vajo.