

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 18. 02. 2011

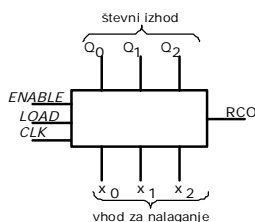
1. Določite minimalno normalno obliko (MNO) za logično funkcijo  $f$  z redundantnimi vhodnimi kombinacijami.

$$f^4 = \sum (0, 1, 6, 7, 14, 15) \text{ in } \sum_x (3, 5, 9, 11, 13)$$

2. Pretvorite podano število  $x$ , zapisano v šestnajstiškem zapisu IEEE754 oblike enojne natančnosti (32 bit single precision) v realno število:

$$x = 3FF00000_{16}$$

3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor z omogočanjem štetja (ENABLE) in vzporednim nalaganjem (LOAD) z D flip-flopi, izbiralniki 2/1 in logičnimi vrati. Podatek se vzporedno nalaga z vhodov ( $x_2, x_1, x_0$ ). Števec naj ima poleg števnege izhoda ( $Q_2, Q_1, Q_0$ ) tudi izhodni prenos za krmiljenje naslednjih stopenj (RCO – ripple carry out). Logika vseh krmilnih signalov je pozitivna. Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol. Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

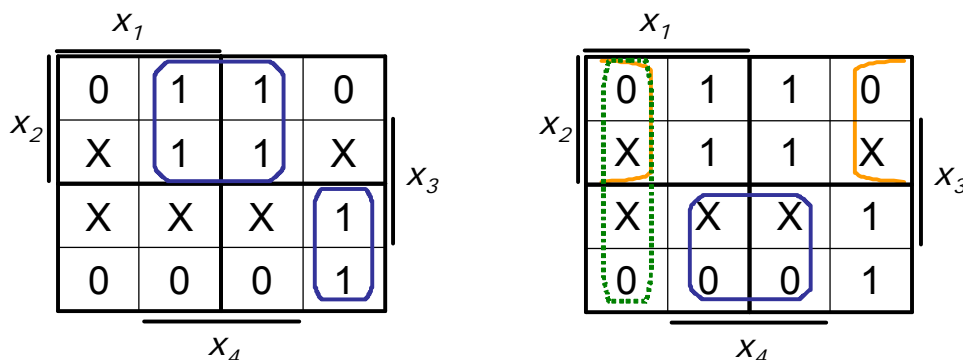
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Funkcijo v PDNO z redundancami moramo izpisati v Veitch–ev diagram, od koder bomo lahko izvajali postopek minimizacije.

$$f^4 = \sum(0,1,6,7,14,15) \text{ in } \sum_x(3,5,9,11,13)$$



Prikazana sta dva Veitch–eva diagrama. Levega uporabljamo za tvorbo MDNO, saj prikazuje funkcijo  $f$ , desnega za tvorbo MKNO, ker združujemo minterme negirane funkcije.

MDNO:

$$f_{MDNO} = x_2 \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_4}$$

Za MKNO pa združujemo ničle (izražamo negirano funkcijo) in izrazimo minimalno obliko negirane funkcije. Nato uporabimo De Morgan–ov teorem, da pridemo do konjunktivne izražave.

$$\begin{aligned} \overline{f} &= x_1 \cdot \overline{x_4} + \overline{x_2} \cdot x_4 + x_2 \cdot \overline{x_4} \\ f &= \overline{x_1 \cdot \overline{x_4} + \overline{x_2} \cdot x_4 + x_2 \cdot \overline{x_4}} \\ f &= \overline{x_1 + x_4} \cdot \overline{x_2 + x_4} \cdot \overline{x_2 + x_4} \\ f_{MKNO} &= (\overline{x_1 + x_4}) \cdot (\overline{x_2 + x_4}) \cdot (\overline{x_2 + x_4}) \end{aligned}$$

Za minimalno normalno obliko moramo določiti, katera izvedba funkcije je cenejša (manjši COST vezja).

OBLIKA	VRAT	VHODOV	COST
MDNO	3	7	10
MKNO	4	9	13

Cenejša izvedba je MDNO, saj je strošek vezja manjši (COST), zato je MNO=MDNO.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 2. naloge: Število  $x$  je podano v IEEE754 zapisu enojne natančnosti.

$$x = 3FF00000_{16}$$

Število izpišemo v dvojiško obliko, tako da vsako šestnajstiško števko pretvorimo v 4 bitno dvojiško število:

3	F	F	0	0	0	0	0
0011	1111	1111	0000	0000	0000	0000	0000

Število razstavimo skladno z obliko IEEE754 (predznak, odmaknjen eksponent, mantisa)

**0 01111111 111000000000000000000000**

Iz bita predznaka sledi, da bo število pozitivno.

Izračunamo še vrednost eksponenta, tako da upoštevamo odmik: Odmaknjen eksponent znaša

**01111111**<sub>2</sub> = **3F**<sub>16</sub> = **127**<sub>10</sub>. Če izračunamo vrednost dejanskega eksponenta odštejemo odmik (127<sub>10</sub>) tako da je dejanski eksponent enak 0.

Mantisi dodamo vodilno enico in decimalno piko v dvojiškem sistemu premaknemo za vrednost eksponenta (v našem primeru ne premikamo pike, ker je eksponent nič).

**1.111000000000000000000000**

Dobljeno vrednost pretvorimo v desetiški sistem.

$$x_2 = 1.111000000000000000000000$$

$$x_{10} = 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + \dots$$

$$x_{10} = 1 + 0.5 + 0.25 + 0.125 = 1.875$$

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

trenutno stanje			naslednje stanje			D–FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Iz tabele prehajanja stanj števca določimo enačbe D–FF:

Za D<sub>0</sub> se iz tabele vidi  $D_0 = Q_0'$

Za D<sub>1</sub> narišemo Veitchev diagram

D<sub>1</sub>:

		<u>Q<sub>2</sub></u>	
<u>Q<sub>1</sub></u>	1	0	0
	0	1	1
		<u>Q<sub>0</sub></u>	

$$D_1 = Q_0 \oplus Q_1$$

Podobno za D<sub>2</sub> narišemo Veitchev diagram

D<sub>2</sub>:

		<u>Q<sub>2</sub></u>	
<u>Q<sub>1</sub></u>	1	0	1
	1	1	0
		<u>Q<sub>0</sub></u>	

Za D<sub>2</sub> sledi:

$$D_2 = Q_2 \cdot Q_1' + Q_2 \cdot Q_0' + Q_2' \cdot Q_1 \cdot Q_0$$

iz česar lahko izpostavimo:

$$D_2 = Q_2 \cdot (Q_1' + Q_0') + Q_2' \cdot Q_1 \cdot Q_0$$

Uporabimo De Morganovo enakost:

$$D_2 = Q_2 \cdot (Q_1 Q_0)' + Q_2' \cdot Q_1 \cdot Q_0$$

iz česar sledi:

$$D_2 = Q_2 \cdot (Q_1 \cdot Q_0)' + Q_2' \cdot Q_1 \cdot Q_0$$

Upoštevamo definicijo XOR operacije ( $a \oplus b = a' \cdot b + a \cdot b'$ )

$$D_2 = Q_2 \oplus Q_1 \cdot Q_0$$

Signal RCO postane '1' takrat, ko števec prešteje do svoje največje vrednosti – v našem primeru postane '1', ko gre stanje števca iz "111" na "000".

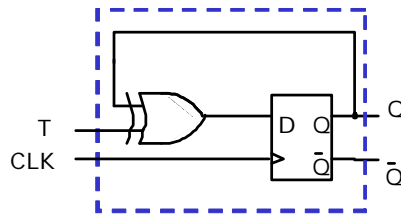
$$RCO = Q_2 \cdot Q_1 \cdot Q_0$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

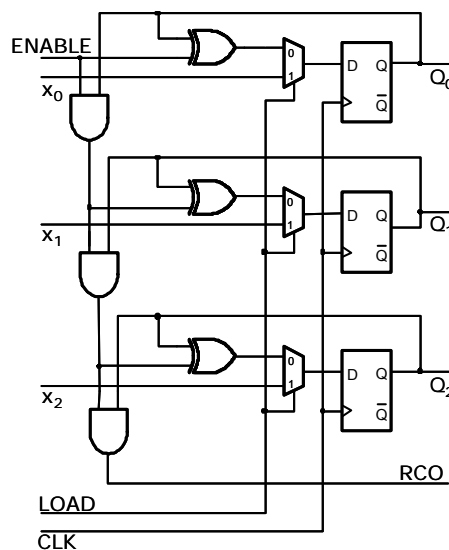
Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (ENABLE). Če vezje analiziramo, vidimo, da smo pravzaprav realizirali T–FF s pomočjo D–FF in XOR vrat, kot kaže spodnja slika:



Slika 1: Realizacija T-FF s pomočjo D-FF.

Če prvemu "T–FF" (D–FF z XOR vrati) postavimo vhod  $T_0 = '0'$  namesto  $T_0 = '1'$ , vsi T–FF ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod  $T_0$  postavimo zunanji signal ENABLE, števec ne bo štel, ampak ohranjal stanje, če bo  $ENABLE = '0'$ . V verigi sinhronega števca so namreč vsi T–FF vezani tako, da so odvisni od prvega T–FF: Če stanje ohranja prvi, ga bodo tudi vsi ostali.

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnege vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame 5 spremenljivk (ENABLE, LOAD,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



Slika 2: Sinhroni števec z vzporednim nalaganjem (LOAD) in omogočanjem štetja (ENABLE) (3–bitna izvedba).

Če želimo z nastalim števcem šteti naraščajoče ... 2, 3, 4, 5, 2, 3, 4, 5 ..., moramo števec, ko le–ta pride do stanja 5 ( $Q_2Q_1Q_0 = 101_2$ ) postaviti nazaj na stanje v stanje 2 ( $Q_2Q_1Q_0 = x_2x_1x_0 = 010_2$ ), torej na LOAD vhod pripeljemo s pomočjo dodatnih dvovhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se ( $Q_2 = 1$  in  $Q_0 = 1$  v števnici sekvenci pojavlja samo enkrat – če bi se večkrat bi morali dekodirati tudi  $Q_1$ ).

Pri tovrstnih števcih ponavadi uporabljamo še zunanji signal RESET, s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod izbiralnikov vodimo LOAD OR RESET.

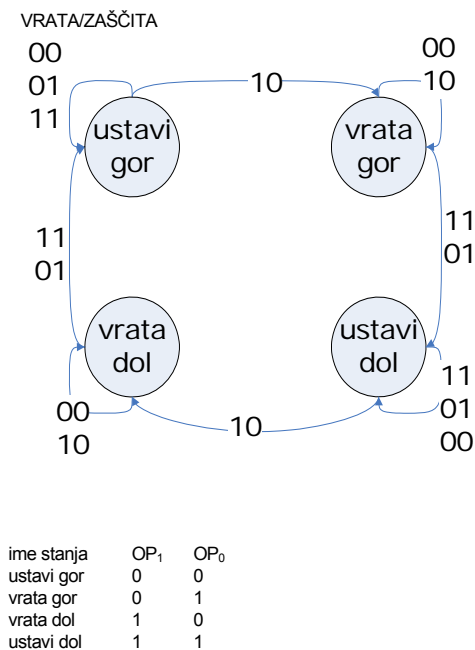
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanja "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji "11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke

VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

Takšna realizacija še zdaleč ni optimalna: Bolje bi bilo, če bi avtomat realizirali kot Mealy-ev tip. Dejanska realizacija ne vsebuje avtomata, temveč en T-FF in relejno logiko.

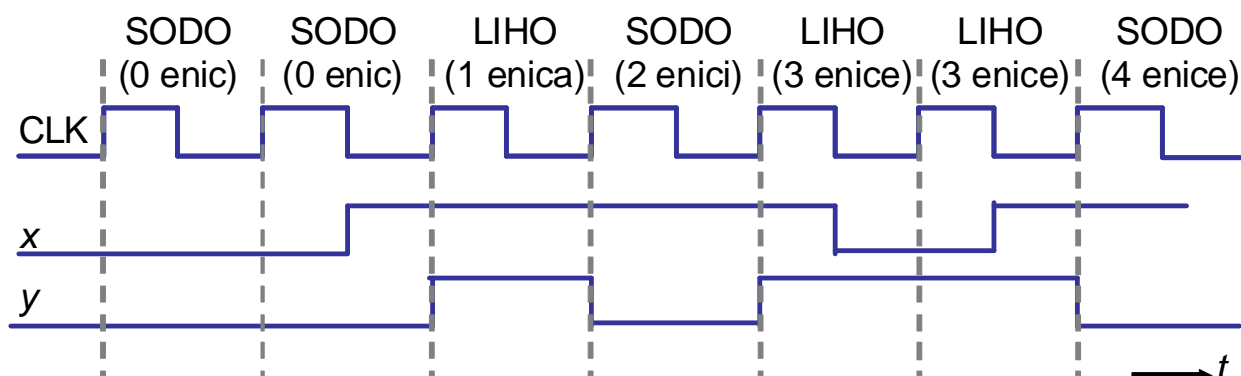
## RAZVOJ DIGITALNIH SISTEMOV

Izpit 15. 04. 2011

1. Zgradite vezje, katerega izhod postane '1', ko se na vhodu pojavi eno izmed praštevil (2, 3, 5, 7, 11, 13). Števila na vhodu so kodirana s 4-bitno Gray-evo kodo. Za realizacijo uporabite samo NAND vrata. Kolikšna je COST funkcija realiziranega vezja?
2. Pretvorite število  $999_{10}$  v dvojiški zapis z uporabo obratnega "double dabble" algoritma.
3. Realizirajte generator lihe parnosti (paritete) kot avtomat končnih stanj, ki šteje število enic v serijskem zaporedju bitov  $x$  na vhodu: Izhod vezja  $y$  naj bo '1', ko je na vhodu liho število enic in '0' ko je na vhodu sodo število enic. Ob resetu avtomata je število enic na vhodu sodo (nič enic). Za realizacijo uporabite D flip-flope, prožene na sprednji rob signala ure  $CLK$ .



Primer delovanja generatorja parnosti povzema spodnja slika:



4. Minimizirajte podani avtomat končnih stanj z uporabo metode z razdelki ter zapišite tabelo prehajanja stanj nastalega minimalnega avtomata.

<i>Trenutno stanje</i>	<i>Naslednje stanje</i>		<i>Izhod</i>
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

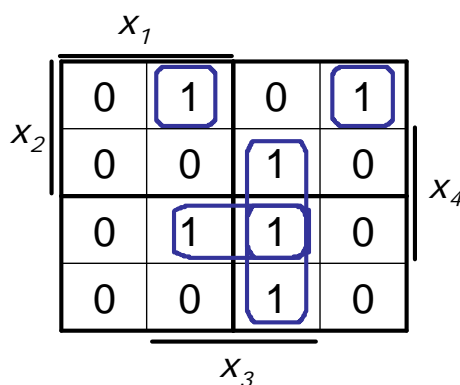
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# Rešitev 1. naloge:

S 4-bitno Grayevo kodo lahko kodiramo 16 števil (0 - 15). Najprej zapišemo pravilnostno tabelo v kateri desetiška števila kodiramo v Grayevi kodi (npr.  $7_{10}=0100_{\text{Gray}}$ ). Izhod mora biti '1' vsakič, ko se na vhodu pojavi praštevilo. Izpisano funkcijo nato izpišemo v Veitch-ev diagram, od koder bomo lahko izvajali postopek minimizacije v PDNO, iz te oblike pa bomo prešli na PSNO (NAND) operatorji.

št.	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	1	1
3	0	0	1	0	1
4	0	1	1	0	0
5	0	1	1	1	1
6	0	1	0	1	0
7	0	1	0	0	1
8	1	1	0	0	0
9	1	1	0	1	0
10	1	1	1	1	0
11	1	1	1	0	1
12	1	0	1	0	0
13	1	0	1	1	1
14	1	0	0	1	0
15	1	0	0	0	0



$$f_{MDNO} = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}$$

Da bi iz PDNO prešli na realizacijo s samimi NAND vrati, funkcijo v PDNO dvakrat negiramo in uporabimo de Morgan-ov teorem:

$$\overline{\overline{f_{MDNO}}} = \overline{\overline{x_1 \cdot \overline{x_2} \cdot x_3 + \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}}}$$

$$f_{PSNO} = \overline{(\overline{x_1 \cdot \overline{x_2} \cdot x_3}) \cdot (\overline{\overline{x_2} \cdot x_3 \cdot x_4}) \cdot (\overline{\overline{x_1} \cdot x_3 \cdot x_4}) \cdot (\overline{x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4}}) \cdot (\overline{\overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}})}$$

Za nastalo PSNO moramo določiti COST vezja (brez inverterjev):

OBLIKA	VRAT	VHODOV	COST
PSNO	6	22	28

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

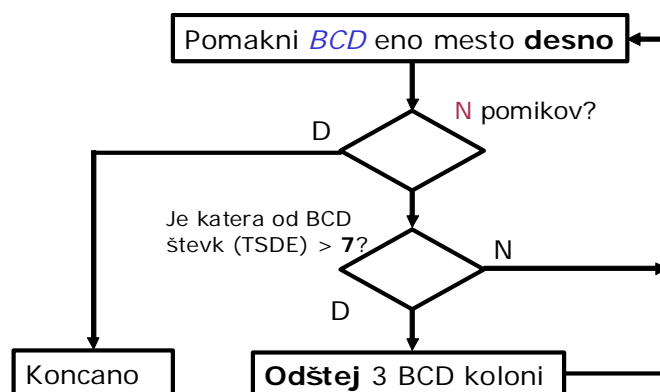
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Rešitev 2. naloge: Če vrednost  $999_{10}$  zapišemo v šestnajstiški obliki dobimo  $3E7_{16}$ .

Postopek pretvorbe poteka po spodaj prikazanem obratnem "Double dabble" algoritmu za pretvorbo števila v BCD zapisu (TSDE) v  $N$  bitno število ( $bin$ ). Začnemo tako, da zapišemo posamezne števice BCD števila:  $1001\ 1001\ 1001_{BCD}$ , jih razdelimo v 3 kolone BCD števk: Štotice, Desetice, Enice in njihove vrednosti po spodnjem algoritmu pomikamo v desno, ter jim odštevamo 3, če je njihova vrednost večja od 7. Po  $N$  pomikih je algoritem končan.



S	D	E	ŠTEVILO	operacija
1 0 0 1	1 0 0 1	1 0 0 1		POMIK1
1 0 0	1 1 0 0	1 1 0 0	1	-3
1 0 0	1 0 0 1	1 0 0 1	1	POMIK2
1 0	0 1 0 0	1 1 0 0	1 1	-3
1 0	0 1 0 0	1 0 0 1	1 1	POMIK3
1	0 0 1 0	0 1 0 0	1 1 1	POMIK4
	1 0 0 1	0 0 1 0	0 1 1 1	-3
	0 1 1 0	0 0 1 0	0 1 1 1	POMIK5
	0 1 1	0 0 0 1	0 0 1 1 1	POMIK6
	0 1	1 0 0 0	1 0 0 1 1 1	-3
	0 1	0 1 0 1	1 0 0 1 1 1	POMIK7
	0	1 0 1 0	1 1 0 0 1 1 1	-3
	0	0 1 1 1	1 1 0 0 1 1 1	POMIK8
		0 0 1 1	1 1 1 0 0 1 1 1	POMIK9
		0 0 1	1 1 1 1 0 0 1 1 1	POMIK10
		0 0	1 1 1 1 1 0 0 1 1 1	POMIK11
		0	0 1 1 1 1 1 0 0 1 1 1	POMIK12
				$3E7_{16}$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da realiziramo avtomat končnih stanj Moore–ove izvedbe:

Najprej bomo razvili diagram prehajanja stanj, ki opisuje delovanje vezja za liho preverjanje paritete. Vezje je lahko v enem od dveh stanj: v sekvenci je bilo do tega trenutka liho ali sodo število enic. Kadar je na vhodu 1, je potrebno preklopiti v drugo stanje. Na primer, če je bilo do tega trenutka prisotnih liho število enic in je trenutni vhod 1, potem bomo imeli sedaj sodo število enic. Če pa bo na vhodu 0, ostane v istem stanju. Narisani diagram prehajanja stanj ima dve stanji, ki označujeta trenutno število enic na vhodu – torej LIHO in SODO. Izhod zapišemo pod stanjem (LIHO='1', SODO='0'). Vrednosti na vhodu  $x$  povzročajo spreminjanje stanj, ki so označene z usmerjenimi povezavami. Če je se na vhodu pojavi '0' (ne glede na to v katerem stanju smo) ostanemo v tem stanju. Jasno – saj štejemo samo '1'. Če smo v stanju LIHO in se na vhodu pojavi '1', preidemo v SODO. Če smo v stanju SODO in se na vhodu pojavi '1', preidemo v LIHO. Povedano povzema spodnji diagram prehajanja stanj

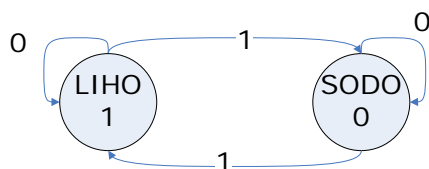


Diagram prehajanja stanj opišemo s tabelo prehajanja stanj:

vhod $x$	trenutno stanje $Q(t)$	naslednje stanje $Q(t+1)$
0	SODO	SODO
0	LIHO	LIHO
1	SODO	LIHO
1	LIHO	SODO

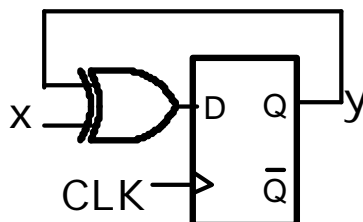
Iz tabele prehajanja stanj avtomata določimo enačbo za D–FF. Potrebno število FF je 1, saj sta stanji samo dve. Iz aplikacijske tabele sledi:

$$D = Q(t+1)$$

$$Q(t+1) = x \cdot \overline{Q(t)} + \overline{x} \cdot Q(t) = x \oplus Q(t)$$

$$y = Q(t)$$

Izvedba vezja je:



Če stanja kodiramo glede na njihov izhod LIHO '1' in SODO '0', potem dobimo novo aplikacijsko tabelo prehajanja stanj.

$x$	$Q(t)$	$Q(t+1)$	$D$	$y$
0	0	0	0	0
0	1	1	1	0
1	0	1	1	1
1	1	0	0	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 4. naloge:

V prvi iteraciji zberemo skupaj vsa stanja v enem razdelku:  $P_1 = (ABCDEFGG)$

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Naslednja iteracija loči stanja, ki imajo različne izhode:  $P_2 = (ABD)(CEFG)$

- Pregledamo vsa naslednja stanja pri vhodu 0 in 1 v vsakem bloku:
    - Blok (ABD):
      - Naslednja stanja pri w=0 (BDB)
      - Naslednja stanja pri w=1 (CFG)
    - Blok (CEFG):
      - Naslednja stanja pri w=0 (FFEF)
      - Naslednja stanja pri w=1 (ECDG)
- Vsa stanja niso v enem bloku. Problem je pri stanju F, ki ima naslednje stanje D. Zato bo stanje F NEEKVIVALENTNO ostalim CEG.
- Novo stanje F zato postavimo v svojo skupino.

Naslednja iteracija loči stanje F od ostalih  $P_3 = (ABD)(CEG)(F)$

- Blok (ABD):
  - Naslednja stanja pri w=0 (BDB)  
So vsa v istem bloku
  - Naslednja stanja pri w=1 (CFG) Niso v istem bloku, ker je F v drugem bloku kot C in G. Zato bo stanje B v novem bloku.
- Blok (CEG):
  - Naslednja stanja pri w=0 (FFF)
  - Naslednja stanja pri w=1 (ECG) C, E in G imamo lahko še vedno za ekvivalentna

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Naslednja iteracija loči stanje B od ostalih  $P_4=(AD)(B)(CEG)(F)$

- Blok (AD)
  - Naslednja stanja pri  $w=0$  (BB)
  - Naslednja stanja pri  $w=1$  (CG)
  - So vsa v istem bloku.
- Blok (CEG)
  - Naslednja stanja pri  $w=0$  (FFF)
  - Naslednja stanja pri  $w=1$  (ECG) So vsa v istem bloku.

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

$P_5=(AD)(B)(CEG)(F)$

Iteraciji  $P_5$  in  $P_4$  sta enaki, zato se postopek minimizacije zaključi. Stanji A in D sta ekvivalentni. Stanja C, E in G so ekvivalentna.

- Tabelo stanj zapišemo na novo
- Izbrišemo vrstice za D, E in G
- Zamenjamo stanja:  $D \rightarrow A$  in vse  $E \rightarrow C$  ter  $G \rightarrow C$

Rezultat je nova tabela stanj minimiziranega avtomata:

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

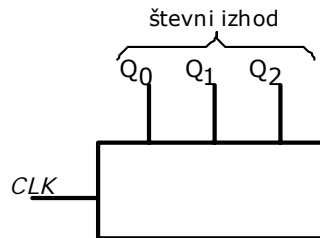
# RAZVOJ DIGITALNIH SISTEMOV

Izpit 21. 06. 2011

1. Ali je funkcija  $f$  linearna? Če je linearna, potem izračunajte koeficiente linearnosti. Če ni linearna, potem utemeljite zakaj.

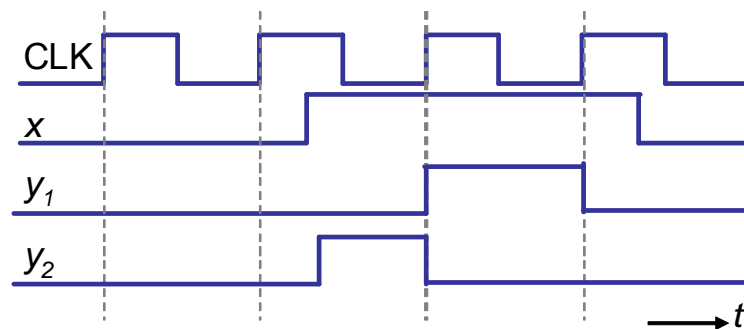
$$f^4 = V(1, 2, 4, 7, 8, 11, 13, 14)$$

2. Pretvorite število  $4B_{16}$  v BCD zapis z uporabo "double dabble" algoritma.
3. Prikažite sintezo 3-bitnega sinhronega števca navzgor po Graye-vi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2, Q_1, Q_0$ ). Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Z uporabo D flip-flopov, ki so proženi na sprednji rob signala ure načrtajte Moore-ov ali Mealy-ev avtomat končnih stanj, ki deluje kot detektor prednjega (pozitivnega) roba prehoda logičnega nivoja vhodnega signala: Avtomat ima vhod  $x$ , izhod  $y$  in signal ure  $CLK$ . Izhod avtomata postane '1' vedno ko se vhodni niz spremeni iz logične '0' na '1'.

Delovanje avtomata povzema spodnja slika. Narisani sta dve realizaciji izhoda  $y_1$  in  $y_2$ : Kateri izhod ( $y_1$  ali  $y_2$ ) pripada Moore-ovemu tip avtomata in kateri Mealy-evemu?



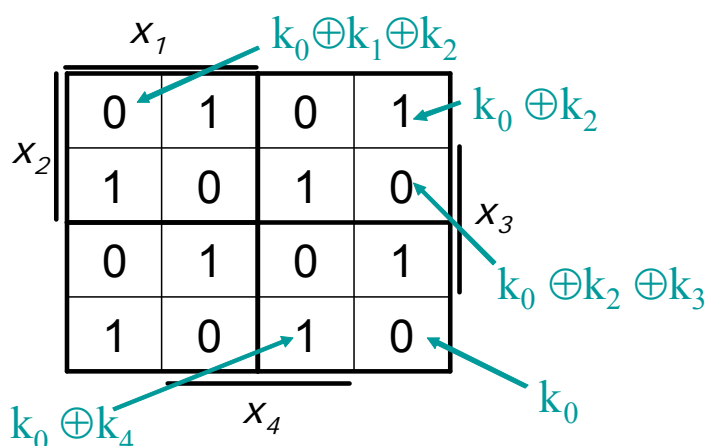
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 1. naloge:

Potrebno je bilo določiti koeficiente linearnosti funkcije podane v PDNO. Linearnost funkcije ugotavljamo tako, da prepognemo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat (recimo da  $x_4$  postane 1 v prvi iteraciji). Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Prepogibanje je prikazano v knjigi, stran 79, vaja 6.5.5.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4 \quad (2.1)$$

S pomočjo Veitch-evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=0$  in  $k_0 \oplus k_4=1$ , kar pomeni  $0 \oplus k_4=1 \rightarrow k_4=1$ .

In če napišemo še eno enačbo za  $k_0 \oplus k_2=1$ , kar pomeni  $0 \oplus k_2=1$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_3=0$ , kar pomeni  $0 \oplus 1 \oplus k_3=0 \rightarrow k_3=1$ .

Če analiziramo naprej dobimo  $k_0 \oplus k_1 \oplus k_2=0$ , kar pomeni  $0 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Če vstavimo dobljeno v enačbo (2.1) dobimo:  $k_0=0 \quad k_1=1 \quad k_2=1 \quad k_3=1 \quad k_4=1$

In rešitev:

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Rešitev 4. naloge:  $4B_{16} = 75_{10}$ . Oziroma zapis posameznih števk: 0111 0101<sub>BCD</sub>.

DESETICE				ENICE				0	1	0	0	1	0	1	1	pomik 1
							0	1	0	0	1	0	1	1		pomik 2
						0	1	0	0	1	0	1	1			pomik 3
					0	1	0	0	1	0	1	1				pomik 4
				0	1	0	0	1	0	1	1					pomik 5
			0	1	0	0	1	0	1	1						+3
			0	1	1	0	0	0	1	1						pomik 6
		0	1	1	0	0	0	1	1							+3
		0	1	1	0	1	1	1	1							pomik 7
	0	1	1	0	1	1	1	1								+3
	0	1	1	1	0	1	0	1								pomik 8
0	1	1	1	0	1	0	1									
7 <sub>10</sub>				5 <sub>10</sub>												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzgor po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
 ...0, 1, 3, 2, 6, 7, 5, 4, 0, 1, 3,...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		Q <sub>2</sub>	
Q <sub>1</sub>	1	0	1
	0	1	0
		Q <sub>0</sub>	

$$T_0 = \overline{Q_2} \oplus Q_1 \oplus Q_0$$

Podobno za T<sub>1</sub> narišemo Veitchev diagram:

		Q <sub>2</sub>	
Q <sub>1</sub>	0	1	0
	0	0	1
		Q <sub>0</sub>	

Iz diagrama za T<sub>1</sub> sledi:

$$T_1 = \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot Q_0$$

$$T_1 = (\overline{Q_2} \cdot \overline{Q_1} + Q_2 \cdot Q_1) \cdot Q_0$$

Operacija v oklepajih je ekvivalenca (negacija XOR), zato enačbo lahko poenostavimo v:

$$T_1 = (\overline{Q_2} \oplus Q_1) \cdot Q_0$$

In še za T<sub>2</sub>:

		Q <sub>2</sub>	
Q <sub>1</sub>	0	0	1
	1	0	0
		Q <sub>0</sub>	

Za T<sub>2</sub> sledi:

$$T_2 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

Operacija v oklepajih je XOR, zato enačbo lahko poenostavimo v:

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Manj potratno možnost realizacije predstavlja navaden dvojiški 3-bitni sinhroni števec navzgor – tega realiziramo s tremi T-FF in enimi AND vrati.

Takemu števcu na izhodu dodamo pretvornik kode iz dvojiškega v Gray-evo kodo z XOR vrati po enačbah:

$$G_{MSB} = B_{MSB}$$

$$G_i = B_{i+1} \oplus B_i$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

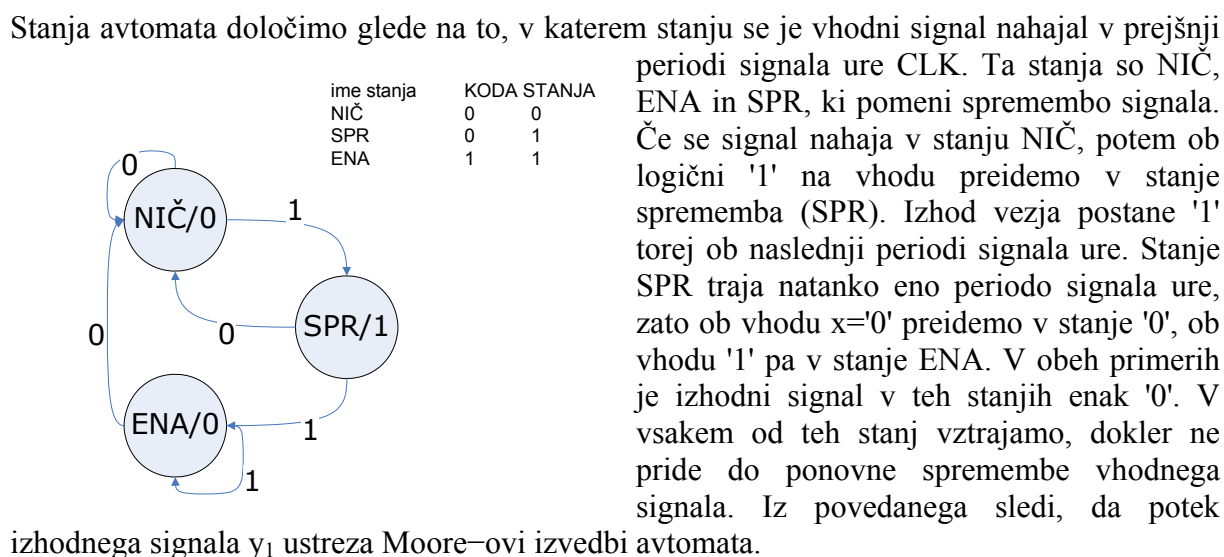
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



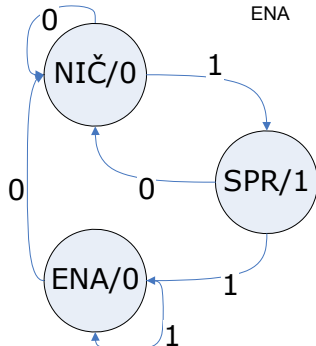
#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj.

Diagram stanj:



ime stanja	KODA STANJA
NIČ	0
SPR	1
ENA	1

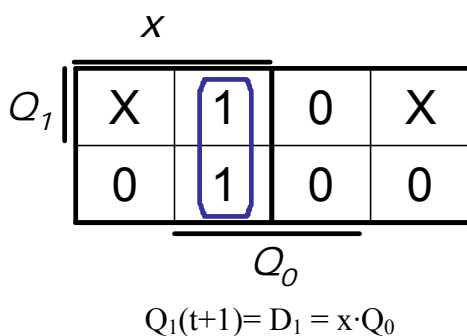


Narišemo tabelo prehajanja stanj

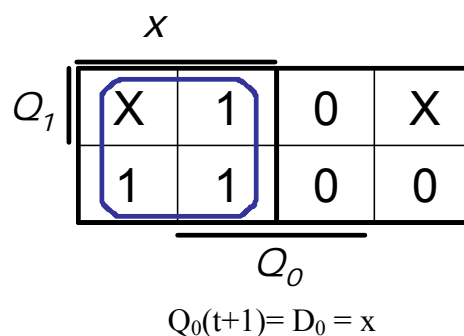
vhod in trenutno stanje			naslednje stanje		enačbe FF in izhoda		
x	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>1</sub>	D <sub>0</sub>	y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	X	X	X	X	X
0	1	1	0	0	0	0	0
1	0	0	0	1	0	1	0
1	0	1	1	1	1	1	1
1	1	0	X	X	X	X	X
1	1	1	1	1	1	1	0

Iz tabele prehajanja stanj avtomata določimo enačbe D-FF:

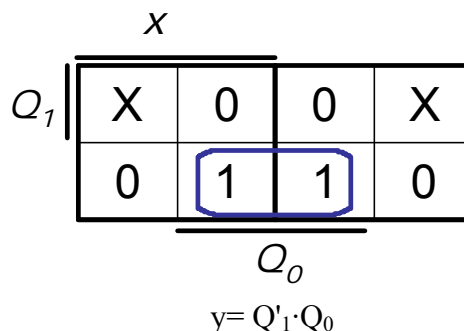
Za Q<sub>1</sub> narišemo Veitchev diagram



Podobno za Q<sub>0</sub> narišemo Veitchev diagram



In še za izhod y:



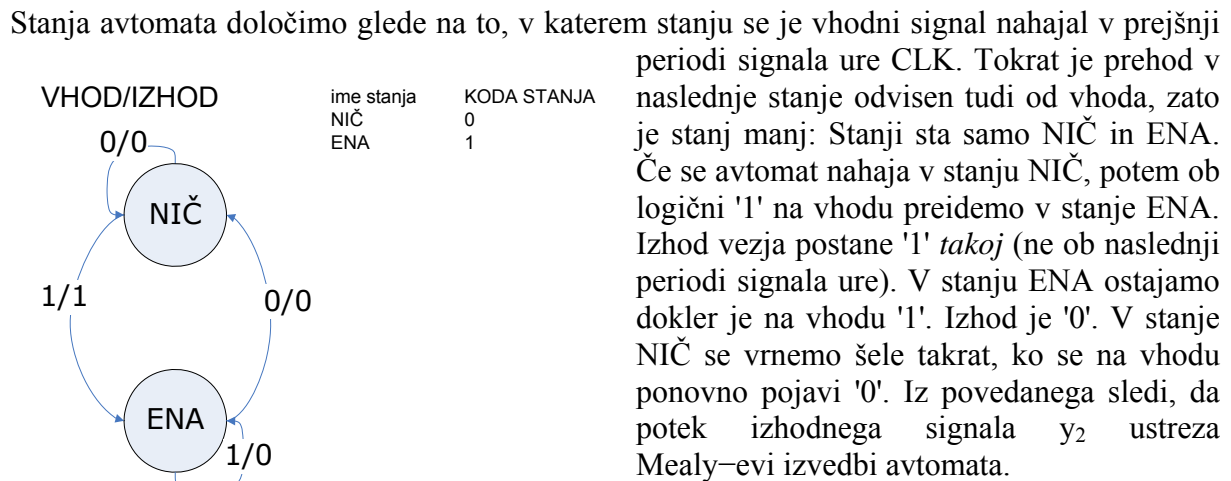
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Mealy–eva realizacija avtomata končnih stanj.

Diagram stanj:



Narišemo tabelo prehajanja stanj:

vhod in trenutno stanje		naslednje stanje	D–FF in izhod	
x	Q	Q	D	y
0	0	0	0	0
0	1	0	0	0
1	0	1	1	1
1	1	1	1	0

Veitchevih diagramov nam ni treba risati, saj enačbi D–FF in izhoda neposredno sledita iz tabele.

$$Q(t+1)=x \quad \text{in} \quad y=x \cdot Q'(t)$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

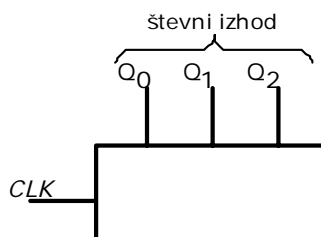
# RAZVOJ DIGITALNIH SISTEMOV

Izpit 9. 9. 2011

1. Z uporabo pravil Boole-ove logike ali Veitch-evih diagramov pokažite, da je logična funkcija linearna. Izračunajte koeficiente linearnosti in jo izrazite v obliki linearnega polinoma.

$$f(x_1, x_2, x_3, x_4) = \overline{x_2} \cdot x_3 \cdot x_4 + x_2 \cdot \overline{x_3} \cdot x_4 + x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$$

2. Pretvorite število  $4B_{16}$  v BCD zapis z uporabo "double dabble" algoritma.
3. Prikažite sintezo 3-bitnega sinhronega števca navzdol po Graye-vi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2$ ,  $Q_1$ ,  $Q_0$ ). Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Z uporabo D flip-flopov, ki so proženi na sprednji rob signala ure CLK, načrtajte Moore-ove avtomat končnih stanj, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov. Krmilje ima:
  - vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov in
  - vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov ter
  - izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

Avtomat ne vrača drobiža in se ob detekciji plačila 15 centov ne vrača nazaj v začetno stanje, ampak ostane v končnem stanju. Vnos dveh kovancev naenkrat ni mogoč.

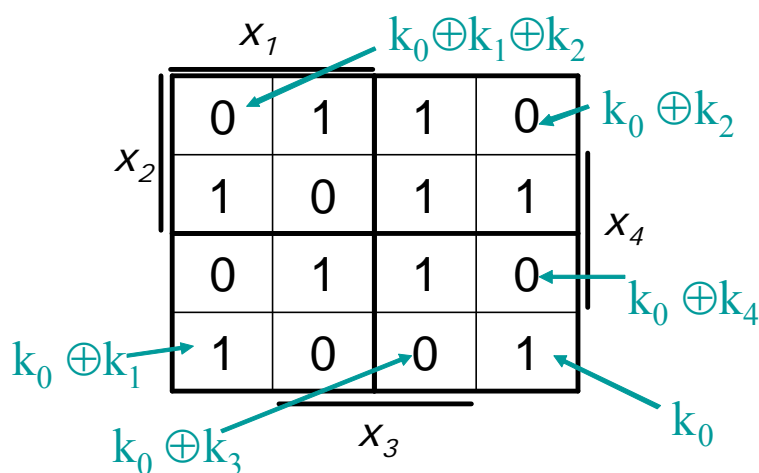
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 1. naloge:

Disjunktivno normalno obliko logične funkcije vpišemo v Veitchev diagram, tako da za vsak konjunktivni izraz poiščemo ustrezne pravokotnike in vanje vpišemo funkcijske vrednosti 1. Prepognemo kvadrat z mintermom  $m_0$  proti  $x_4$  in nato oba skupaj proti spremenljivki  $x_3$ . Rezultat prepogiba je negacija vrednosti, kar izpolnjuje pogoj linearnosti, zato prepognemo štiri kvadrate proti spremenljivki  $x_2$ . Rezultat prepogiba je negacija v vseh štirih kvadratih, zato nadaljujemo z zadnjim prepogibom osmih kvadratov proti  $x_1$ , kjer imamo enakost. Pregledali smo cel Veitchev diagram in ker smo povsod dobili izpolnjen pogoj enakosti ali popolne negacije pomeni, da je logična funkcija linearna. Prepogibanje je prikazano v knjigi, stran 79, vaja 6.5.5.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4 \quad (2.1)$$

S pomočjo Veitch-evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_4=0$ , kar pomeni  $1 \oplus k_4=0 \rightarrow k_4=1$ .

Iz enačbe  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

Če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $0 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_1=1$ , kar pomeni  $1 \oplus k_1=0 \rightarrow k_1=0$ .

In končna rešitev:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_2 \oplus x_3 \oplus x_4$$

Do istega pridemo lahko z uporabo pravil Boole-ove logike. Izhodišče je podana funkcija.

$$f(x_1, x_2, x_3, x_4) = \overline{x_2} \cdot x_3 \cdot x_4 + x_2 \cdot \overline{x_3} \cdot x_4 + x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$$

Izvedemo razčlenitev po  $x_4$ :

$$f(x_1, x_2, x_3, x_4) = (\overline{x_2} \cdot x_3 + x_2 \cdot \overline{x_3}) \cdot x_4 + (x_2 \cdot x_3 + \overline{x_2} \cdot \overline{x_3}) \cdot \overline{x_4}$$

Člena v oklepajih predstavljata XOR operacijo in pa EQU operacijo. Za dve spremenljivki velja da je XOR negacija ekvivalence.

$$\begin{aligned} x \oplus y &= \overline{x} \cdot y + x \cdot \overline{y} \\ x \equiv y &= \overline{x \oplus y} = \overline{x} \cdot \overline{y} + x \cdot y \end{aligned}$$

Zgornji enakosti vpišemo v enačbo za funkcijo  $f$ , s tem da ekvivalenco raje pišemo kot negacijo XOR funkcije.

$$f(x_1, x_2, x_3, x_4) = (x_2 \oplus x_3) \cdot x_4 + \overline{(x_2 \oplus x_3)} \cdot \overline{x_4}$$

Uvedemo novo spremenljivko  $g$ :

$$g = x_2 \oplus x_3$$

Funkcijo zapišemo kot:

$$f(x_1, x_2, x_3, x_4) = g \cdot x_4 + \overline{g} \cdot \overline{x_4}$$

Od koder sledi

$$f(x_1, x_2, x_3, x_4) = \overline{g \oplus x_4}$$

Iz zgornje enačbe sledi, da imamo negacijo XOR funkcije, kar lahko zopet izrazimo z XOR funkcijo kot:

$$\overline{x} = 1 \oplus x$$

Če z obratnim vstavljanjem izrazimo funkcijo  $f$  s spremenljivkami  $x_1..x_4$  dobimo končen rezultat:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_2 \oplus x_3 \oplus x_4$$

Od tod bi podobno lahko prebrali koeficiente:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4 \\ f(x_1, x_2, x_3, x_4) &= 1 \oplus 0 \cdot x_1 \oplus 1 \cdot x_2 \oplus 1 \cdot x_3 \oplus 1 \cdot x_4 \end{aligned}$$

Rešitev 2. naloge:  $4B_{16} = 75_{10}$ . Oziroma zapis posameznih števk: 0111 0101<sub>BCD</sub>.

DESETICE				ENICE				0	1	0	0	1	0	1	1	pomik 1
							0	1	0	0	1	0	1	1		pomik 2
						0	1	0	0	1	0	1	1			pomik 3
					0	1	0	0	1	0	1	1				pomik 4
				0	1	0	0	1	0	1	1					pomik 5
			0	1	0	0	1	0	1	1						+3
			0	1	1	0	0	0	1	1						pomik 6
		0	1	1	0	0	0	1	1							+3
		0	1	1	0	1	1	1	1							pomik 7
	0	1	1	0	1	1	1	1								+3
	0	1	1	1	0	1	0	1								pomik 8
0	1	1	1	0	1	0	1									
$7_{10}$				$5_{10}$												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzdol po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
 ... 0, 4, 5, 7, 6, 2, 3, 1, 0, ...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1
0	1	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	0	1

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		$Q_2$			
$Q_1$	0	1	0	1	
	1	0	1	0	
		$Q_0$			

$$T_0 = Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za T<sub>1</sub> narišemo Veitchev diagram

		$Q_2$			
$Q_1$	0	0	1	0	
	0	1	0	0	
		$Q_0$			

Za T<sub>1</sub> sledi:

$$T_1 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_1 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

Operacija v oklepajih je XOR, zato enačbo lahko poenostavimo v:

$$T_1 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

In še za T<sub>2</sub>:

		$Q_2$			
$Q_1$	1	0	0	0	
	0	0	0	1	
		$Q_0$			

Za T<sub>2</sub> sledi:

$$T_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot \overline{Q_1} + Q_2 \cdot Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR (za dve spremenljivki je to ekvivalenca), zato enačbo lahko poenostavimo v:

$$T_2 = (\overline{Q_2 \oplus Q_1}) \cdot \overline{Q_0}$$

Manj potratno možnost realizacije predstavlja navaden dvojiški 3-bitni sinhroni števec navzgor – tega realiziramo s tremi T-FF in enimi AND vrati.

Takemu števcu na izhodu dodamo pretvornik kode iz dvojiškega v Gray-evo kodo z XOR vrati po enačbah:

$$G_{MSB} = B_{MSB}$$

$$G_i = B_{i+1} \oplus B_i$$

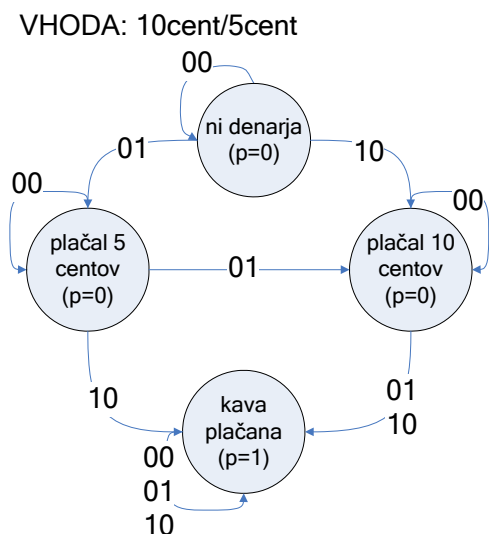
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj. Opis diagrama stanj:



vrgel bo izhod v teh dveh stanjih enak  $p=0$ , ker še ni plačal celotne cene kave. Če smo v stanju "plačal 5 centov" in uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ). Stanje "kava plačana" je končno in tam tudi ostanemo za vse možne kombinacije. Če smo v stanju "plačal 10 centov" in uporabnik vrže v avtomat 5 ali 10 centov (10cent/5cent=10 oz. 01), potem podobno preidemo v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ).

Naredimo tabelo prehajanja stanj:

trenutno stanje	10cent	5cent	naslednje stanje	izhod p
ni denarja	0	0	ni denarja	0
ni denarja	0	1	plačal 5 centov	0
ni denarja	1	0	plačal 10 centov	0
ni denarja	1	1	X	X
plačal 5 centov	0	0	plačal 5 centov	0
plačal 5 centov	0	1	plačal 10 centov	0
plačal 5 centov	1	0	kava plačana	0
plačal 5 centov	1	1	X	X
plačal 10 centov	0	0	plačal 10 centov	0
plačal 10 centov	0	1	kava plačana	0
plačal 10 centov	1	0	kava plačana	0
plačal 10 centov	1	1	X	X
kava plačana	0	0	kava plačana	1
kava plačana	0	1	kava plačana	1
kava plačana	1	0	kava plačana	1
kava plačana	1	1	X	X

Izberemo kodiranje stanj:

stanje	$Q_1$	$Q_0$
ni denarja	0	0
plačal 5 centov	0	1
plačal 10 centov	1	0
kava plačana	1	1

Na začetku se nahajamo v stanju "ni denarja", v katerem je izhod  $p=0$ . Vhoda v avtomat sta dva: 10cent in 5cent, kar na diagramu kodiramo kot 10cent/5cent.

Mehanizem za vnos kovancev preprečuje hkraten vnos dveh kovancev, torej je kombinacija (10cent/5cent=11) nemogoča, zato bo avtomat od tu lahko prešel v poljubno stanje (X). Če uporabnik ni vrgel denarja v avtomat (10cent/5cent=00), potem ostaja v stanju "ni denarja". Če uporabnik vrže v avtomat 5 centov (10cent/5cent=01), potem preide v stanje "plačal 5 centov". Če uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "plačal 10 centov". Ne glede na to koliko je

Nad tabelo prehajanja stanj uporabimo predlagano kodiranje stanj:

$Q_1$	$Q_0$	10cent	5cent	$Q_1$	$Q_0$	izhod p
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	X	X	X
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	X	X	X

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Naloga zahteva realizacijo z D–FF:

t				t+1				
Q <sub>1</sub>	Q <sub>0</sub>	10cent	5cent	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>1</sub>	D <sub>0</sub>	izhod p
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	1	X	X	X	X	X
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	1	1	0
0	1	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0	0
1	0	0	1	1	1	1	1	0
1	0	1	0	1	1	1	1	0
1	0	1	1	X	X	X	X	X
1	1	0	0	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	X	X	X	X	X

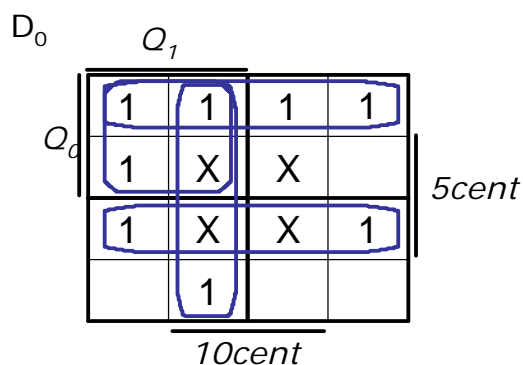
Iz dobljene tabele narišemo Veitch–eve diagrame za oba D–FF in izhod p:

$$D_0 = V(1, 4, 6, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$D_1 = V(2, 5, 6, 8, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$p = V(12-14) \text{ in } Vx(3, 7, 11, 15)$$

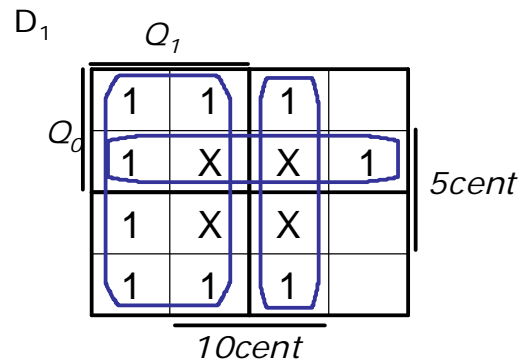
Veitch–ev diagram za D<sub>0</sub>:



$$D_0 = Q_1 \cdot Q_0 + \overline{5cent} \cdot Q_0 + 10cent \cdot Q_1 + 5cent \cdot \overline{Q_0}$$

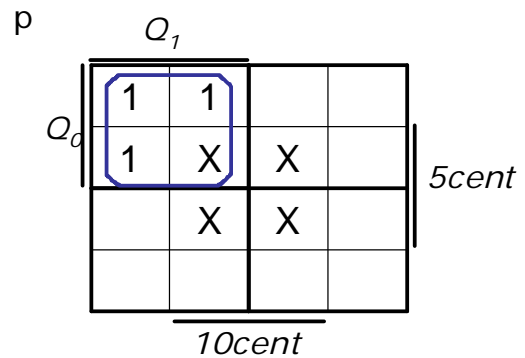
$$D_0 = Q_1 \cdot (Q_0 + 10cent) + 5cent \oplus Q_0$$

Veitch–ev diagram za D<sub>1</sub>:



$$D_1 = Q_1 + 5cent \cdot Q_0 + 10cent$$

Veitch–ev diagram za izhod p:



$$p = Q_1 \cdot Q_0$$

Enačbo za izhod p bi lahko napisali tudi samo s sklepanjem, saj se izhod p postavi samo, ko je avtomat v stanju "kava plačana", ki ima kodo Q<sub>1</sub>Q<sub>0</sub>="11" – torej ko bosta Q<sub>1</sub> in Q<sub>0</sub> enaka '1', bo izhod p=1. Iz dobljenih enačb minimizacije bi lahko narisali vezje krmilja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 23. 9. 2011

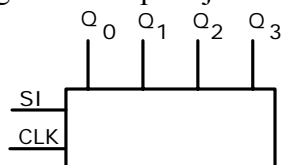
1. Z uporabo pravil Boole-ove logike zapišite logično funkcijo v KNO z NAND (Sheffer-jevimi) operatorji.

$$f(x_1, x_2, x_3) = (x_1 + x_2) \cdot (\overline{x_2} + \overline{x_3}) \cdot x_3$$

2. Narišite strukturo vezja PAL3L3 (izmišljeno vezje) in ga uporabite za realizacijo funkcij:

- $f_1 = x_1 \oplus x_2$
  - $f_2$  = konjunkcijo treh spremenljivk
  - $f_3$  = funkcijo treh spremenljivk, ki vrne 1 pri vsaj dveh enicah na vhodih.
- Vsaka disjunkcija (OR) v PAL3L3 ima 4 konjunkcije (AND). Oznaka L pomeni, da je vezje tipa AND–NOR. Povezave označite s piko (●).

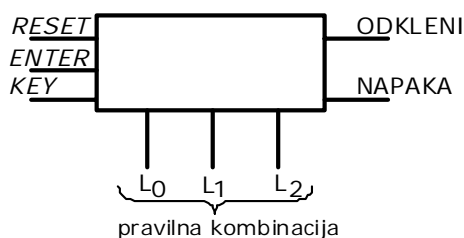
3. Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod  $SI$  (ang. serial input), in vzporedni izhod ( $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ ). Uporabite poimenovanje signalov na spodnji sliki.



4. Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev (*RESET*), ki postavlja ključavnico v začetno stanje, tipko (*ENTER*) za vnos nastavljenе kombinacije in preklopnik (*KEY*) za vnos enega bita kombinacije. Izhod ključavnice je (*ODKLENI*), ki postane '1' ko uporabnik vnesе pravilno kombinacijo in izhod (*NAPAKA*), ki postane '1' če je vnesena kombinacija napačna. Uporabo ključavnice povzema spodnje zaporedje:

- 1.) pritisnemo *RESET*
- 2.) s preklopnikom *KEY* nastavimo bit kombinacije odklepanja
- 3.) pritisnemo *ENTER*
- 4.) izvede se primerjava i-tega bita ( $KEY \equiv L_i$ )
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi *ODKLENI*='1', sicer se postavi *NAPAKA*='1'.
- 7.) ponoven pritisk na *RESET* nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti  $L_0$ ,  $L_1$  in  $L_2$ .



1. Z uporabo pravil Boole-ove logike moramo zapisati logično funkcijo v KNO z NAND (Sheffer-jevimi) operatorji:

Direktne poti za pretvorbo KNO v zapis s NAND (Sheffer-jevimi) operatorji ni, zato KNO najprej pretvorimo v DNO obliko, tako da izvedemo AND operacije nad dvočleniki:

$$f(x_1, x_2, x_3) = (x_1 + x_2) \cdot (\overline{x_2} + \overline{x_3}) \cdot x_3$$

V drugem členu bomo dobili člen  $x \cdot x' = 0$  po lastnostih Boole-ove algebre:

$$f(x_1, x_2, x_3) = (x_1 \cdot \overline{x_2} + x_1 \cdot \overline{x_3} + x_2 \cdot \overline{x_3}) \cdot x_3$$

S preostalimi členi izpišemo AND operacije:

$$f(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_3} \cdot x_3 + x_2 \cdot \overline{x_3} \cdot x_3$$

Podobno pri drugem in tretjem členu funkcije dobimo  $x \cdot x' = 0$ :

$$f(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot x_3$$

Nastala funkcija ima samo en člen, ki ga moramo izraziti z NAND operatorji. Funkcijo negiramo in dobimo:

$$\overline{f(x_1, x_2, x_3)} = \overline{x_1 \cdot \overline{x_2} \cdot x_3}$$

Končno izrazimo še negacijo funkcije  $f$ :

$$\overline{x} = 1 \uparrow x$$

Nazadnje izrazimo original funkcije  $f$ , kot zahteva naloga:

$$f(x_1, x_2, x_3) = 1 \uparrow (x_1 \uparrow \overline{x_2} \uparrow x_3)$$

2. Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2 =$  konjunkcija treh spremenljivk
- $f_3 =$  funkcijo treh spremenljivk, ki vrne 1 pri vsaj dveh enicah na vhodih.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 produkte (AND). L pa pomeni, da je izhod negiran. Povezave oz. ‘varovalke’ označite s piko (•).

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$f_2 = \overline{x_1 \cdot x_2 \cdot x_3}$$

$$\overline{f_2} = \overline{x_1} + \overline{x_2} + \overline{x_3}$$

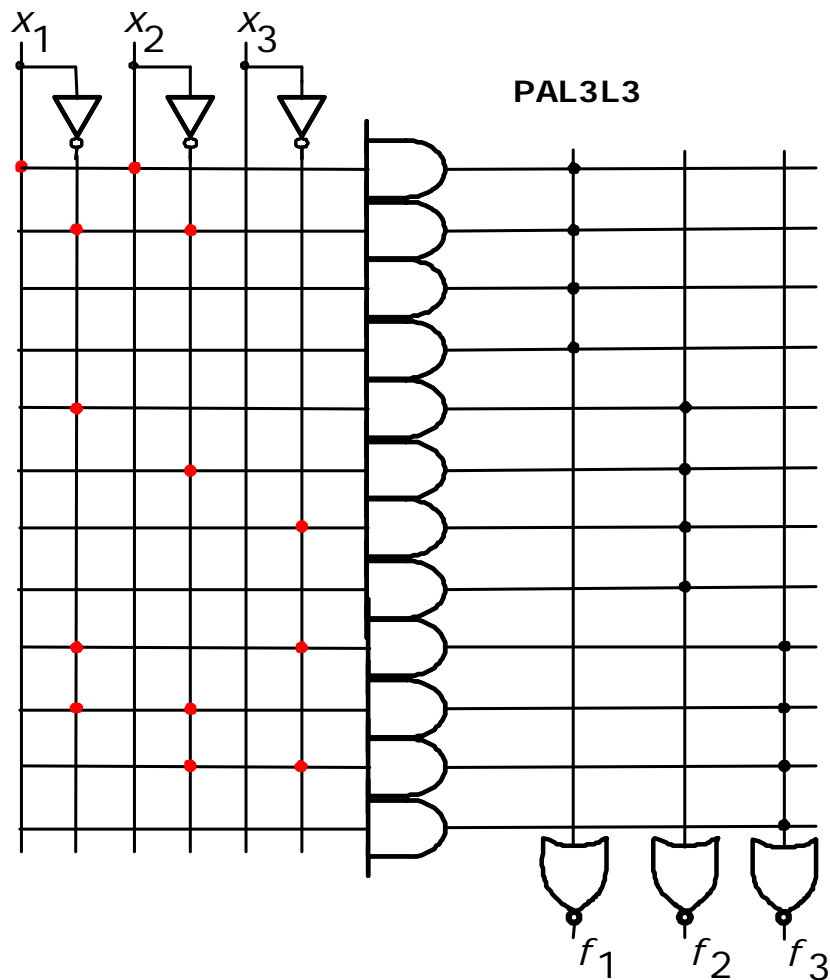
Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

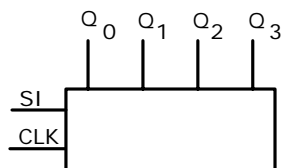
$$\overline{f_3} = \overline{x_1} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vezemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6-vhodna. Vezje PAL3L3 je AND-NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.

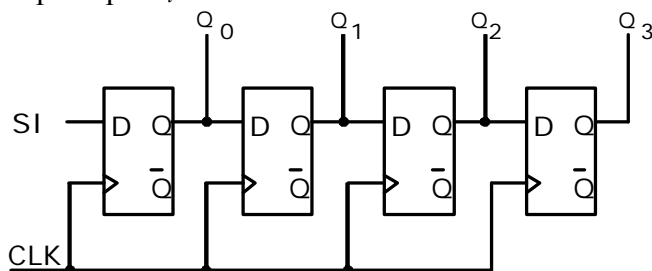


Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

3. Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod  $SI$  (ang. serial input), in vzporedni izhod ( $Q_0, Q_1, Q_2, Q_3$ )



Zaporedno-vzporedni (SIPO) pomikalni register, realiziran s pomočjo D-FF, je veriga kaskadno vezanih D-FF, v kateri je izhod prejšnjega flip-flopa  $Q_{i-1}$  vezan na vhod naslednjega flip-flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz T-FF in 2/1 izbiralnikov, moramo pravzaprav realizirati celico D-FF s pomočjo T-FF in 2/1 izbiralnikov. V ta namen zapišemo tabelo D-FF, pri kateri dodamo izhodni stolpec  $T$  vhoda.

$D$	$Q(t)$	$Q(t+1)$	$T$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

XOR vrata moramo realizirati s pomočjo 2/1 izbiralnikov, zato zapišemo enačbo XOR funkcije:

$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

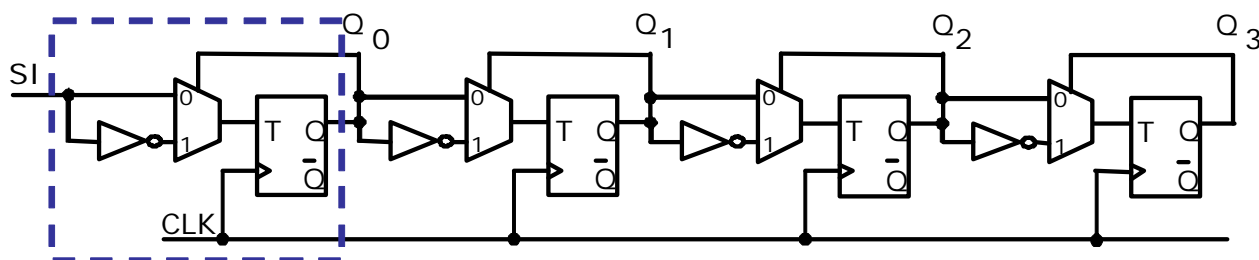
Iz tabele sledi, da je  $T$  vhod XOR operacija  $Q(t)$  in vhoda D-FF, ki ga realiziramo.

Funkcijo  $f$  realiziramo z izbiralnikom tako, da naredimo razvoj po spremenljivki  $x$  in dobimo:

$$Q(t+1) = Q(t) \oplus D$$

$x$	$f$
0	$y$
1	$y'$

Če nastali D-FF iz T-FF in 2/1 izbiralnika sestavimo skupaj v 4-bitni pomikalni register dobimo spodnjo realizacijo, v kateri je izvedba D-FF označena črtkano.

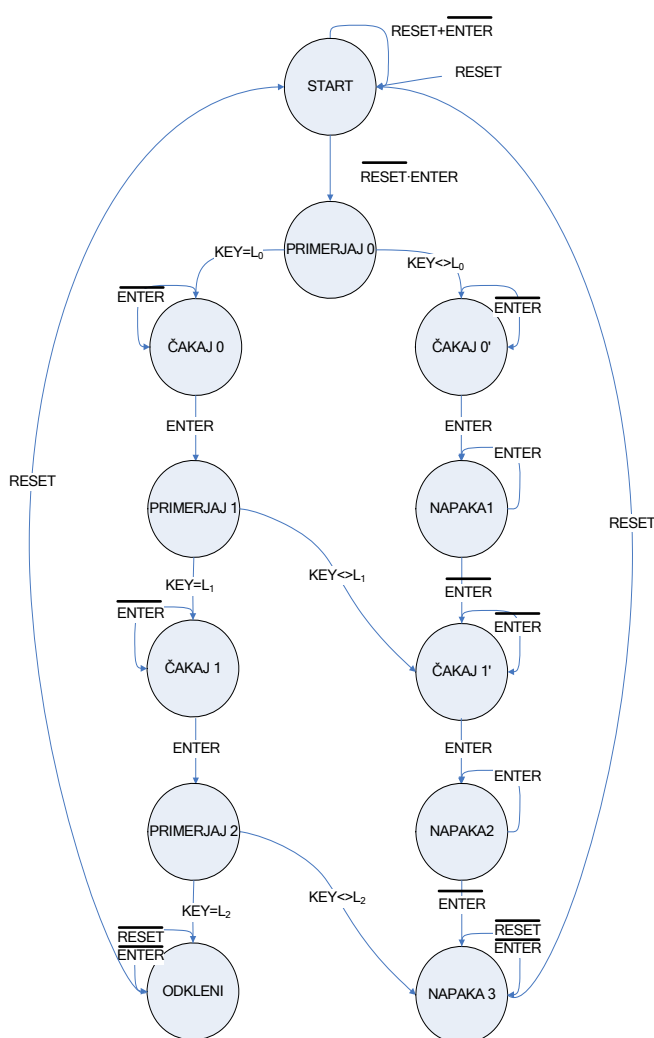
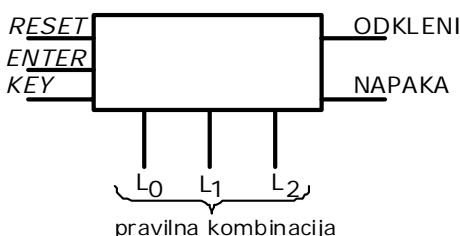


4. Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev (*RESET*), ki postavlja ključavnico v začetno stanje, tipko (*ENTER*) za vnos nastavljene kombinacije in preklopnik (*KEY*) za vnos enega bita kombinacije. Izhod ključavnice je (*ODKLENI*), ki postane '1' ko uporabnik vnese pravilno kombinacijo in izhod (*NAPAKA*), ki postane '1' če je vnesena kombinacija napačna.

Uporabo ključavnice povzema spodnje zaporedje:

- 1.) pritisnemo *RESET*
- 2.) s preklopnikom *KEY* nastavimo bit kombinacije odklepanja
- 3.) pritisnemo *ENTER*
- 4.) izvede se primerjava i-tega bita ( $KEY = L_i$ )
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi *ODKLENI*='1', sicer se postavi *NAPAKA*='1'.
- 7.) ponoven pritisk na *RESET* nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti  $L_0$ ,  $L_1$  in  $L_2$ .



Ključavnica se ob vklopu ali ob ponastavitvi (*RESET*) nahaja v stanju *START*. V tem stanju uporabnik nastavi preklopnik *KEY* v stanje prvega bita kombinacije ('0' ali '1'). Iz tega stanja lahko pride v stanje *PRIMERJAJ 0* samo ob pogoju, da *RESET* ni pritisnjen in da je *ENTER* pritisnjen. V tem stanju je lahko vnesena kombinacija pravilna ( $KEY = L_0$ ) ali nepravilna ( $KEY \neq L_0$ ). Če je pravilna, potem preide v stanje *ČAKAJ 0*, v katerem čaka da uporabnik spusti tipko *ENTER*. Uporabnik v tem stanju nastavi drugi bit kombinacije (*KEY*) in ponovno pritisne *ENTER*. Avtomat preide v stanje

*PRIMERJAJ 1*, od koder sta zopet dve možnosti. Omenjeni postopek se lahko ponavlja za več bitov kombinacije. Bistveno je, da pred vsako primerjavo postavimo stanje čakanja, v katerem čakamo, da uporabnik spusti tipko *ENTER*, nato nastavi bit kombinacije in šele nato preide v stanje nove primerjave ob ponovnem pritisku na *ENTER*. V zadnjem stanju primerjave avtomat preide v stanje *ODKLENI*.

Če se uporabnik pri vnašanju zmoti, preide avtomat v sekvenco stanj napake, po kateri mora vnesti še dve mesti kode (lahko samo dvakrat pritisne *ENTER*) in šele nato preide v stanje *NAPAKA 3*, v katerem se postavi izhod *NAPAKA*.



# RAZVOJ DIGITALNIH SISTEMOV

Izpit 26. 01. 2012

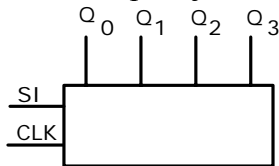
1. Pretvorite število  $3E7_{16}$  v BCD zapis z uporabo "double dabble" algoritma.

2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2$  = konjunkcijo treh spremenljivk
- $f_3$  = funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko.

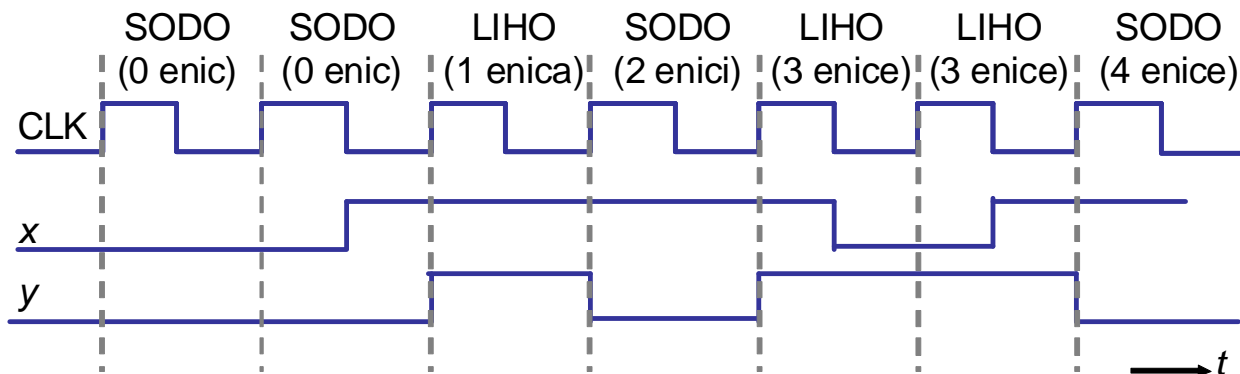
3. Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod *SI* (ang. serial input), in vzporedni izhod ( $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ ). Uporabite poimenovanje signalov na spodnji sliki.



4. Realizirajte generator lihe parnosti (paritete) kot avtomat končnih stanj, ki šteje število enic v serijskem zaporedju bitov *x* na vhodu: Izhod vezja *y* naj bo '1', ko je na vhodu liho število enic in '0' ko je na vhodu sodo število enic. Ob resetu avtomata je število enic na vhodu sodo (nič enic). Za realizacijo uporabite D flip-flope, prožene na sprednji rob signala ure *CLK*.



Primer delovanja generatorja parnosti povzema spodnja slika:



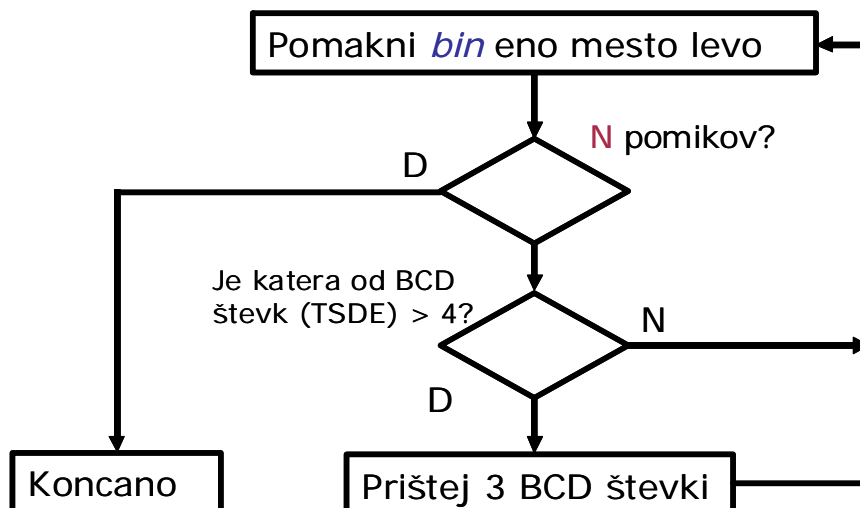
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:  $3E7_{16} = 999_{10}$ . Zapis posameznih števk rezultata:  $1001\ 1001\ 1001_{BCD}$ .

Postopek pretvorbe poteka po spodaj prikazanem "Double dabble" algoritmu za pretvorbo  $N$  bitnega števila ( $bin$ ) v BCD zapis (TSDE)



STOTICE				DESETICE				ENICE				<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>0</i>	<i>0</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	
											1	1	1	1	1	0	0	1	1	1			
										1	1	1	1	1	0	0	1	1	1				
									1	1	1	1	1	1	0	0	1	1	1				
								1	0	1	0	1	1	1	0	0	1	1	1				
							1	0	1	0	1	1	0	0	1	1	1						
							1	1	0	0	0	1	0	0	1	1	1						
						1	1	0	0	0	1	0	0	1	1	1							
						1	1	0	0	0	1	0	0	1	1	1							
					1	0	0	1	0	0	1	0	0	1	1	1							
			1	0	0	1	0	0	1	0	0	1	1	1									
		1	0	0	0	1	0	0	1	0	0	1	1										
		1	0	0	0	1	0	0	1	1	0	0	1	1									
	1	0	0	0	1	0	0	1	1	0	0	1	1										
	1	0	0	0	1	1	0	0	1	1	0	0	1	1									
1	0	0	1	1	0	0	1	1	0	0	1												
9 <sub>10</sub>				9 <sub>10</sub>				9 <sub>10</sub>															

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\overline{f_2} = \overline{x_1 \cdot x_2 \cdot x_3}$$

$$\overline{f_2} = \overline{x_1} + \overline{x_2} + \overline{x_3}$$

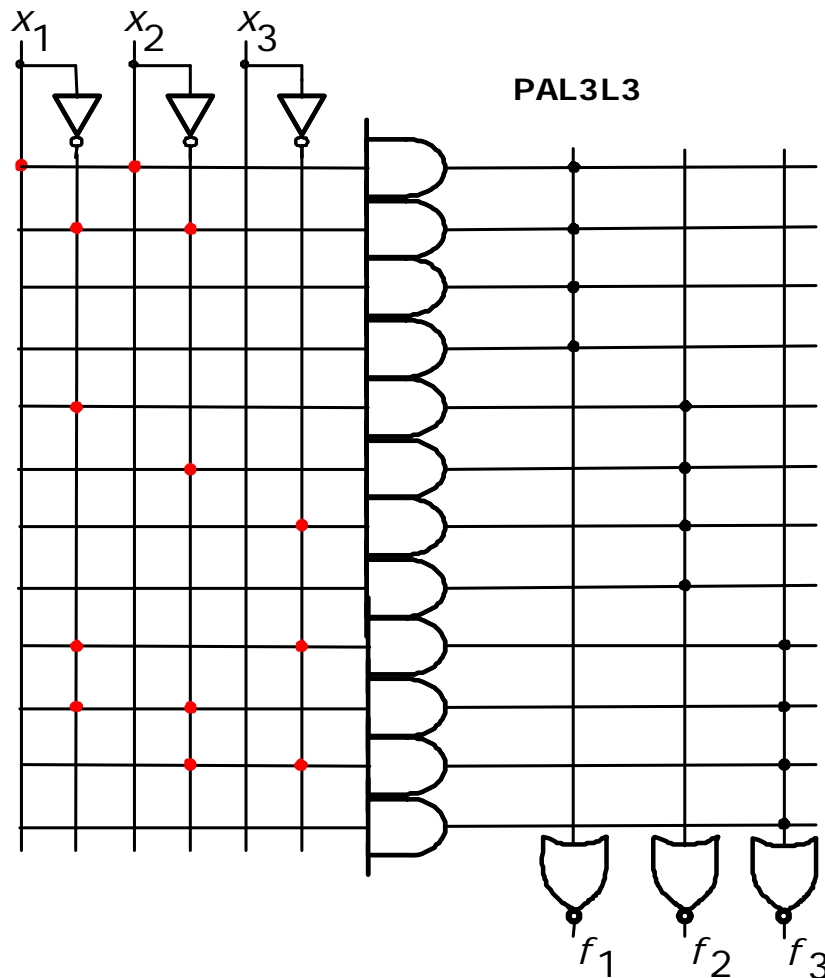
Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

$$\overline{f_3} = \overline{x_1 \cdot x_3} + \overline{x_1 \cdot x_2} + \overline{x_2 \cdot x_3}$$

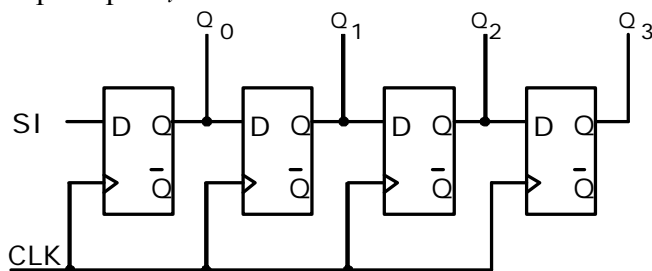
Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vežemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6-vhodna. Vezje PAL3L3 je AND-NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

### Rešitev 3. naloge:

Zaporedno–vzporedni (SIPO) pomikalni register, realiziran s pomočjo D–FF, je veriga kaskadno vezanih D–FF, v kateri je izhod prejšnjega flip–flopa  $Q_{i-1}$  vezan na vhod naslednjega flip–flopa  $D_i$ .



realiziramo.

Če želimo pomikalni register sestaviti iz T–FF in 2/1 izbiralnikov, moramo pravzaprav realizirati celico D–FF s pomočjo T–FF in 2/1 izbiralnikov. V ta namen zapišemo tabelo D–FF, pri kateri dodamo izhodni stolpec  $T$  vhoda.

$$Q(t+1) = Q(t) \oplus D$$

XOR vrata moramo realizirati s pomočjo 2/1 izbiralnikov, zato zapišemo enačbo XOR funkcije:

$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

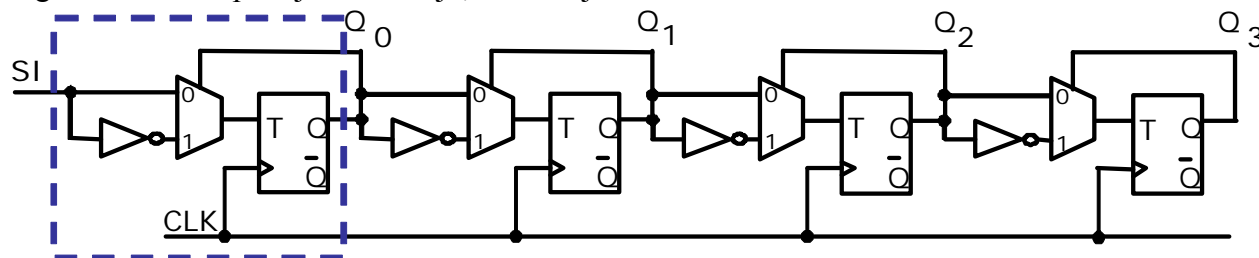
$D$	$Q(t)$	$Q(t+1)$	$T$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Funkcijo  $f$  realiziramo z izbiralnikom tako, da naredimo razvoj po spremenljivki  $x$  in dobimo:

$x$	$f$
0	$y$
1	$y'$

Iz tabele sledi, da je  $T$  vhod XOR operacija  $Q(t)$  in vhoda D–FF, ki ga

Če nastali D–FF iz T–FF in 2/1 izbiralnika sestavimo skupaj v 4–bitni pomikalni register dobimo spodnjo realizacijo, v kateri je izvedba D–FF označena črtkano.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Postopek sinteze zahteva, da realiziramo avtomat končnih stanj Moore-ove izvedbe. Najprej bomo razvili diagram prehajanja stanj, ki opisuje delovanje vezja za liho preverjanje parnosti. Vezje je lahko v enem od dveh stanj: v sekvenci je bilo do tega trenutka liho ali sodo število enic. Kadar je na vhodu 1, je potrebno preklopiti v drugo stanje. Na primer, če je bilo do tega trenutka prisotnih liho število enic in je trenutni vhod 1, potem bomo imeli sedaj sodo število enic. Če pa bo na vhodu 0, ostane v istem stanju. Narisani diagram prehajanja stanj ima dve stanji, ki označujeta trenutno število enic na vhodu – torej LIHO in SODO. Izhod zapišemo pod stanjem (LIHO='1', SODO='0'). Vrednosti na vhodu  $x$  povzročajo spreminjanje stanj, ki so označene z usmerjenimi povezavami. Če je se na vhodu pojavi '0' (ne glede na to v katerem stanju smo) ostanemo v tem stanju: Jasno – saj štejemo samo '1'. Če smo v stanju LIHO in se na vhodu pojavi '1', preidemo v SODO. Če smo v stanju SODO in se na vhodu pojavi '1', preidemo v LIHO. Povedano povzema spodnji diagram prehajanja stanj

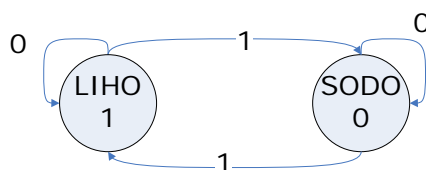


Diagram prehajanja stanj opišemo s tabelo prehajanja stanj:

vhod $x$	trenutno stanje $Q(t)$	naslednje stanje $Q(t+1)$
0	SODO	SODO
0	LIHO	LIHO
1	SODO	LIHO
1	LIHO	SODO

Če stanja kodiramo glede na njihov izhod LIHO '1' in SODO '0', potem dobimo novo aplikacijsko tabelo prehajanja stanj.

$x$	$Q(t)$	$Q(t+1)$	$D$	$y$
0	0	0	0	0
0	1	1	1	0
1	0	1	1	1
1	1	0	0	1

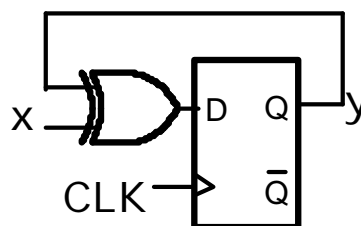
Iz tabele prehajanja stanj avtomata določimo enačbo za D-FF. Potrebno število FF je 1, saj sta stanji samo dve. Iz aplikacijske tabele sledi:

$$D = Q(t+1)$$

$$Q(t+1) = x \cdot \overline{Q(t)} + \overline{x} \cdot Q(t) = x \oplus Q(t)$$

$$y = Q(t)$$

Izvedba vezja je:



Realizirali smo T-FF, prožen na sprednji rob signala ure.

Delovanje vezja si lahko ogledate v predlogah Logisim na domači strani predmeta:

Logisim\ff\T\_ff\_using\_D\_ff\_and\_xor.circ

# RAZVOJ DIGITALNIH SISTEMOV

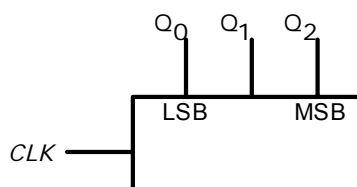
Izpit 27. 03. 2012

1. Realizirajte funkcijo  $f$  s čim manj izbiralniki 4/1.

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Pretvorite število  $59_{16}$  v BCD zapis z uporabo "double dabble" algoritma.

3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor po Graye-vi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2, Q_1, Q_0$ ) in vhod za signal ure (CLK). Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol. Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

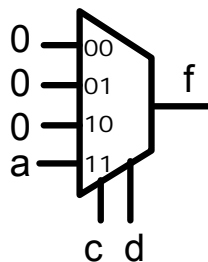
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

$$f_{PDNO}(a,b,c,d) = V(11,15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).

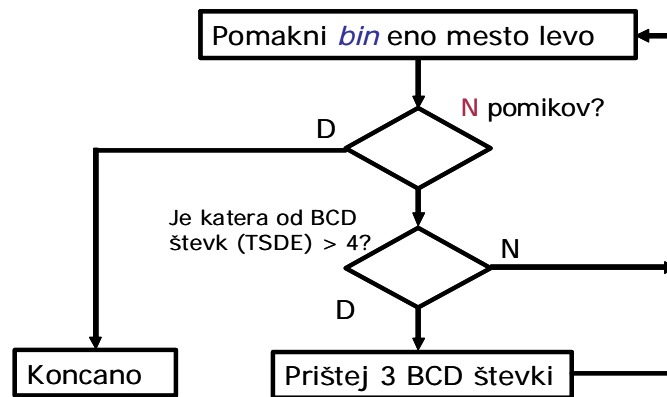




Rešitev 2. naloge:

Pretvorimo šestnajstiko vrednost  $59_{16} = 89_{10}$ . Zapis posameznih števk:  $0101\ 1001_{BCD}$ .

Pretvorbo po "double dabble" algoritmu opravimo po spodnjem algoritmu:



DESETICE				ENICE				0	1	0	1	1	0	0	1	pomik 1
							0	1	0	1	1	0	0	1		pomik 2
						0	1	0	1	1	0	0	1			pomik 3
					0	1	0	1	1	0	0	1				pomik 4
				0	1	0	1	1	0	0	1					pomik 5
				1	0	0	0	1	0	0	1					+3
			1	0	0	0	1	0	0	1						pomik 6
		1	0	0	0	1	0	0	1							pomik 7
	1	0	0	0	1	0	0	1								pomik 8
1	0	0	0	1	0	0	1									
$8_{10}$				$9_{10}$												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzgor po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
 ... 0, 1, 3, 2, 6, 7, 5, 4, 0...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		Q <sub>2</sub>			
		Q <sub>1</sub>		Q <sub>0</sub>	
T <sub>0</sub>	Q <sub>1</sub>	1	0	1	0
	Q <sub>0</sub>	0	1	0	1

$$T_0 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{Q_2} \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \cdot \overline{Q_0} + Q_1 \cdot Q_0) + Q_2 \cdot (\overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0})$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \oplus \overline{Q_0}) + Q_2 \cdot (Q_1 \oplus Q_0)$$

Uvedemo novo spremenljivko x:

$$x = Q_1 \oplus Q_0$$

in jo vstavimo v izraz za T<sub>0</sub>:

$$T_0 = \overline{Q_2} \cdot \overline{x} + Q_2 \cdot x = \overline{Q_2} \oplus x$$

$$T_0 = 1 \oplus Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za T<sub>1</sub> narišemo Veitchev diagram:

		Q <sub>2</sub>			
		Q <sub>1</sub>		Q <sub>0</sub>	
T <sub>1</sub>	Q <sub>1</sub>	0	1	0	0
	Q <sub>0</sub>	0	0	1	0

Iz diagrama za T<sub>1</sub> sledi:

$$T_1 = Q_2 \cdot Q_1 \cdot Q_0 + \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0$$

$$T_1 = (Q_2 \cdot Q_1 + \overline{Q_2} \cdot \overline{Q_1}) \cdot Q_0$$

Operacija v oklepajih je ekvivalenca, zato enačbo lahko poenostavimo v:

$$T_1 = (\overline{Q_2} \oplus Q_1) \cdot Q_0$$

Podobno storimo še za T<sub>2</sub>:

		Q <sub>2</sub>			
		Q <sub>1</sub>		Q <sub>0</sub>	
T <sub>2</sub>	Q <sub>1</sub>	0	0	0	1
	Q <sub>0</sub>	1	0	0	0

Iz diagrama za T<sub>2</sub> sledi:

$$T_2 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR, zato enačbo lahko poenostavimo v:

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Manj potratno možnost realizacije predstavlja dvojiški 3-bitni sinhroni števec navzgor. Tak števec realiziramo s tremi T-FF in enimi AND vrati.

Nastalemu sinhronemu števcu na izhodu dodamo pretvornik kode iz dvojiškega v Gray-evo kodo z XOR vrati po enačbah:

$$G_{MSB} = B_{MSB}$$

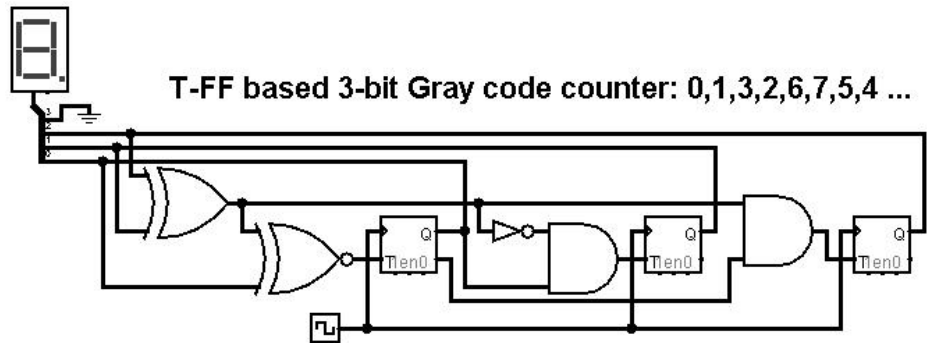
$$G_i = B_{i+1} \oplus B_i$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

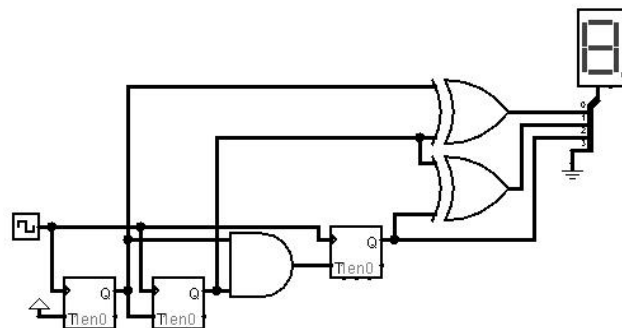
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revB.circ



Enostavnejšo izvedbo štetja dosežemo z uporabo sinhronnega 3 bitnega dvojiškega števca in pretvornika kode iz dvojiške v Gray–evo kodo.

**T-FF based 3-bit Gray code counter: 0,1,3,2,6,7,5,4 ...**



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revC.circ

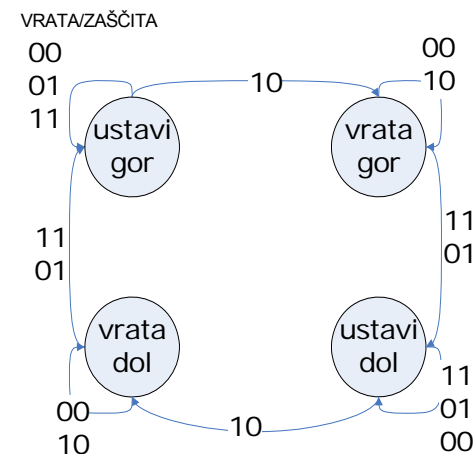
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



ime stanja	OP <sub>1</sub>	OP <sub>0</sub>
ustavi gor	0	0
vrata gor	0	1
vrata dol	1	0
ustavi dol	1	1

Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanja "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji "11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke

VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 13. 09. 2012

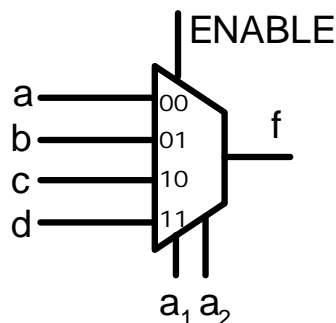
1. Določite popolno konjunktivno normalno obliko (PKNO) in popolno disjunktivno normalno obliko (PDNO) funkcije  $f$ .

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

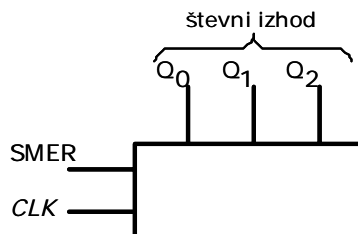
2. Realizirajte funkcijo  $f^4 = V(3,6,7,10,15)$  z redundantnimi mintermi pri  $V_x(1,12,13)$  z enim izbiralnikom 4/1, ki ima vhod "ENABLE".

Delovanje izbiralnika 4/1 z vhodom "ENABLE" povzema spodnja tabela:

ENABLE	$a_1$	$a_2$	$f$
0	X	X	0
1	0	0	a
1	0	1	b
1	1	0	c
1	1	1	d



3. Prikažite sintezo sinhronega dvosmernega 3-bitnega števca z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov. Števec ima vhod SMER, ki določa smer štetja: Če je SMER='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Minimizirajte podani avtomat končnih stanj z uporabo metode z razdelki ter zapišite tabelo prehajanja stanj nastalega minimalnega avtomata.

Trenutno stanje	Naslednje stanje		Izhod
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

## Rešitev 1. naloge

Funkcija je zapisana v večnivojski obliki, torej jo izrazimo v normalno obliko.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

Funkciji NOR ( $\downarrow$ ) in ekvivalence ( $\equiv$ ) izpišemo:

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 + x_2}) \cdot \overline{x_3} + \left( \overline{(\overline{x_2 \oplus x_4}) + x_1} \right)$$

Ekvivalenco smo izrazili kot negacijo XOR. Uporabimo De Morganov teorem:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2 \oplus x_4}) \cdot x_1$$

Izpišemo enačbo funkcije XOR ( $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$ ) in dobimo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2} \cdot \overline{x_4} + x_2 \cdot x_4) \cdot x_1$$

Razširimo še zadnjo konjunkcijo in rezultat je oblika MDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_4} \cdot x_1 + x_2 \cdot x_4 \cdot x_1$$

Če uporabimo lastnost Boole-ove algebre ( $\overline{\overline{a}} + a = 1$ ) lahko zapišemo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}$$

Kar lahko zapišemo v obliki PDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4$$
$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$

PDNO pretvorimo v PKNO tako, da pregledamo manjkajoče minterme: 2, 3, 4, 5, 6, 7, 9, 11, 12, 14. Te minterme preslikamo preko tabele:

m <sub>i</sub>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M <sub>i</sub>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

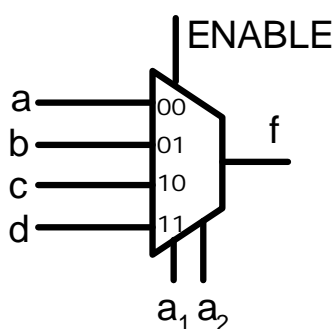
Funkcija v PKNO se torej glasi:

$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$
$$f_{PKNO}(x_1, x_2, x_3, x_4) = \&(13, 12, 11, 10, 9, 8, 6, 4, 3, 1)$$

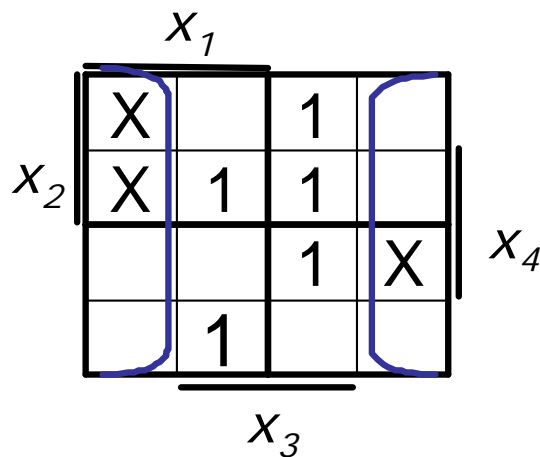
## Rešitev 2. naloge:

Delovanje izbiralnika 4/1 z vhomom "ENABLE" povzema spodnja tabela:

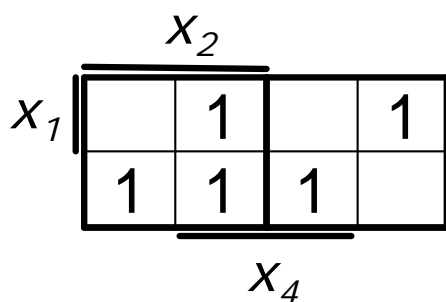
ENABLE	$a_1$	$a_2$	$f$
0	X	X	0
1	0	0	a
1	0	1	b
1	1	0	c
1	1	1	d



Funkcijo narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

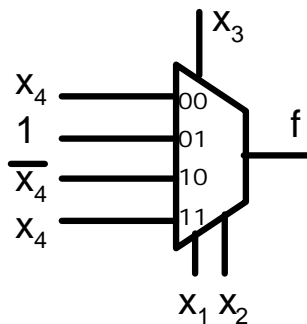


Čim imamo na voljo izbiralnik z ENABLE vhomom preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '0' vključno z redundancami X. V zgornjem Veitch–evem diagramu je to spremenljivka  $x_3$ : Namreč, če je  $x_3=0$ , potem lahko vse redundance izberemo tako, da bo  $f=0$  za vse vrednosti  $x_3=0$ . Ko to spremenljivko določimo, narišemo samo tisti del Veitch–evega diagrama štirih spremenljivk, pri katerem bo  $x_3=1$ .



$a_1$	$a_2$	$x_1x_2$	$x_1x_4$	$x_2x_4$
0	0	$x_4$	$x_2$	$x_1$
0	1	1	1	$x_1'$
1	0	$x_4'$	$x_2'$	$x_1'$
1	1	$x_4$	$x_2$	1

Nastali Veitch–ev diagram razvijemo po vseh možnih kombinacijah dveh spremenljivk in rezultat zapišemo v tabeli. Vse realizacije so enako komplicirane, tako da je vseeno katero realiziramo. Odločimo se za realizacijo po  $x_1x_2$ .



### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF:

Iz stolpca T<sub>0</sub> se vidi, da je T<sub>0</sub>=1'. Iz stolpca T<sub>1</sub> se vidi, da se ponavlja vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T <sub>1</sub>
0	Q <sub>0</sub>
1	Q <sub>0</sub> '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

Za T<sub>2</sub> se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

	SMER				
Q <sub>2</sub>	1	0	0	0	Q <sub>0</sub>
	0	0	1	0	
	0	0	1	0	
	1	0	0	0	
Q <sub>1</sub>					

$$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$$

V enačbi za T<sub>2</sub> poiščemo podobnosti z enačbo za T<sub>1</sub>: Enačba za T<sub>1</sub> vsebuje konjunkciji SMER·Q<sub>0</sub>' in SMER'·Q<sub>0</sub>, ki sta vsebovani tudi v enačbi za T<sub>2</sub>, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191<sup>1</sup>. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.

Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

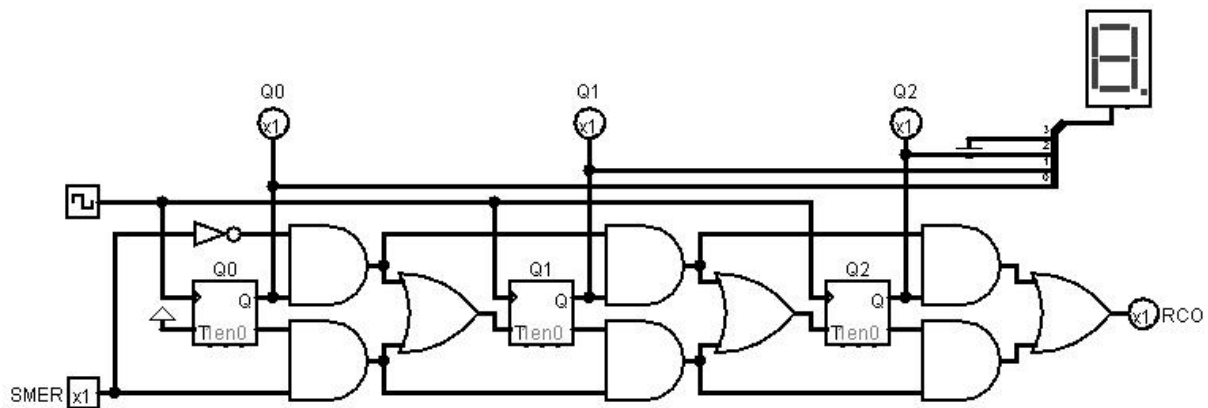
<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74191>



Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca (RCO – oz. ripple carry out, včasih tudi TC – terminal count). RCO je signal, ki postane '1' ob prehodu iz najvišjega stanja števca (v našem primeru je to "111") v stanje "000" pri štetju navzgor in ob prehodu "000" v najvišje stanje števca pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števce vežemo kaskadno – torej da signal RCO vežemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu  $Q_2$ , kot kaže spodnja slika:



Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_up\_down\_3\_bit\_using\_T\_FF.circ

Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

- desetiški (BCD) števcji, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161<sup>2</sup>.

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74161>

Rešitev 4. naloge:

V prvi iteraciji zberemo skupaj vsa stanja v enem razdelku:  $P_1 = (ABCDEFGG)$

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Naslednja iteracija loči stanja, ki imajo različne izhode:  $P_2 = (ABD)(CEFG)$

- Pregledamo vsa naslednja stanja pri vhodu 0 in 1 v vsakem bloku:
    - Blok (ABD):
      - Naslednja stanja pri w=0 (BDB)
      - Naslednja stanja pri w=1 (CFG)
    - Blok (CEFG):
      - Naslednja stanja pri w=0 (FFEF)
      - Naslednja stanja pri w=1 (ECDG)
- Vsa stanja niso v enem bloku. Problem je pri stanju F, ki ima naslednje stanje D. Zato bo stanje F NEEKVIVALENTNO ostalim CEG.
- Novo stanje F zato postavimo v svojo skupino.

Naslednja iteracija loči stanje F od ostalih  $P_3 = (ABD)(CEG)(F)$

- Blok (ABD):
  - Naslednja stanja pri w=0 (BDB)  
So vsa v istem bloku
  - Naslednja stanja pri w=1 (CFG) Niso v istem bloku, ker je F v drugem bloku kot C in G. Zato bo stanje B v novem bloku.
- Blok (CEG):
  - Naslednja stanja pri w=0 (FFF)
  - Naslednja stanja pri w=1 (ECG) C, E in G imamo lahko še vedno za ekvivalentna

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Naslednja iteracija loči stanje B od ostalih  $P_4=(AD)(B)(CEG)(F)$

- Blok (AD)
  - Naslednja stanja pri  $w=0$  (BB)
  - Naslednja stanja pri  $w=1$  (CG)
  - So vsa v istem bloku.
- Blok (CEG)
  - Naslednja stanja pri  $w=0$  (FFF)
  - Naslednja stanja pri  $w=1$  (ECG) So vsa v istem bloku.

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

$P_5=(AD)(B)(CEG)(F)$

Iteraciji  $P_5$  in  $P_4$  sta enaki, zato se postopek minimizacije zaključi. Stanji A in D sta ekvivalentni. Stanja C, E in G so ekvivalentna.

- Tabelo stanj zapišemo na novo
- Izbrišemo vrstice za D, E in G
- Zamenjamo stanja:  $D \rightarrow A$  in vse  $E \rightarrow C$  ter  $G \rightarrow C$

Rezultat je nova tabela stanj minimiziranega avtomata:

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

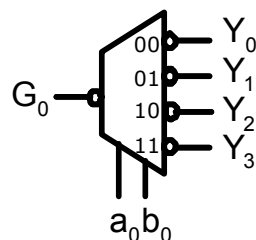
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 13. 06. 2012

- Realizirajte funkcijo  $f^4 = V(1, 2, 4, 7)$  z redundantnimi mintermi pri  $V_x(0, 3, 6)$  z enim TTL dekodermem 74139. Dekoder 74139 ima vhod za omogočenje elementa ( $G$ ) in izhode  $Y_0, Y_1, Y_2, Y_3$  v negativni logiki. Njegovo delovanje povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

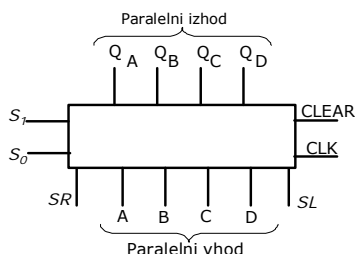


- Uporabite ROM vezje za realizacijo naslednjih funkcij:

$$g_1 = x_1 + x_2 \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2$$

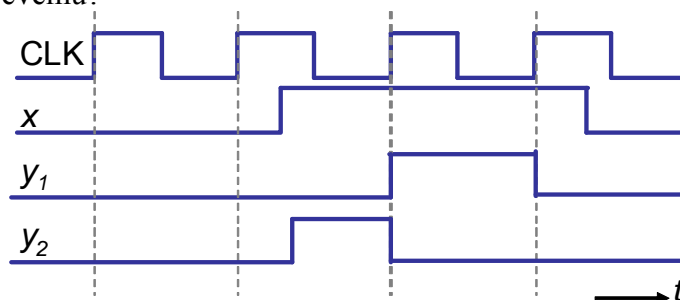
ROM vezje ima 3 naslovne spremenljivke in 2 bitno vsebino. Narišite shemo elementa in v shemi označite programirane povezave oz. 'varovalke' s piko (●).

- Z uporabo D flip-flopov, ki so proženi na pozitivni rob signala ure  $CLK$ , prikažite sintezo univerzalnega 4-bitnega pomikalnega registra, ki ima dva funkcijska vhoda  $S_0$  in  $S_1$  in opravlja funkcije po spodnji tabeli: Register ima tudi zaporedna vhoda za pomik v levo (SL – serial left), pomik v desno (SR – serial right) in asinhroni vhod za brisanje CLEAR (aktiven nizek).



$S_1$	$S_0$	funkcija
0	0	drži stanje
0	1	pomik vsebine eno mesto desno
1	0	pomik vsebine eno mesto levo
1	1	nalaga vsebino z vhodov ABCD

- Z uporabo D flip-flopov, ki so proženi na sprednji rob signala ure ( $CLK$ ) načrtajte Mooreov ali Mealyev avtomat končnih stanj, katerega izhod  $y$  postane '1' ko se vhod  $x$  spremeni iz logične '0' na '1'. Delovanje avtomata povzema spodnja slika. Narisani sta dve realizaciji izhoda avtomata  $y_1$  in  $y_2$ : Kateri izhod ( $y_1$  ali  $y_2$ ) pripada Moore-ovemu tip avtomata in kateri Mealy-evemu?



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

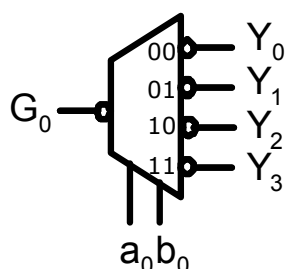
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Delovanje dekoderja 74139<sup>1</sup> povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Funkcijo  $f$  narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

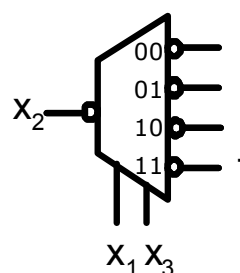
	$x_1$			
$x_2$	X	1	X	1
	1	0	1	X
	$x_3$			

Čim imamo na voljo dekodirnik z ENABLE vhomom v negativni logiki preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '1' vključno z redundancami X.

V zgornjem Veitch–evem diagramu je to spremenljivka  $x_2$ : Namreč, če je  $x_2=1$ , potem lahko vse redundance izberemo tako, da bo  $f=1$  za vse vrednosti  $x_2=1$ . Ko določimo spremenljivko za omogočenje elementa ( $G$ ), opazujemo samo preostali del Veitch–evega diagrama. Spodnje 4 vrednosti diagrama narišemo v novem Veitch–evem diagramu 2 spremenljivk.

	$x_1$	
$x_3$	0	1
	1	X

Dekoder ima aktivno nizke izhode, zato iz nastalega diagrama realiziramo *negacijo* funkcije  $f$  zato v Veitch–evem diagramu združujemo ničle, kar nastopa samo v primeru ko sta  $x_1=1$  in  $x_3=1$ .



<sup>1</sup><http://www.alldatasheet.com/view.jsp?Searchword=74HC139>

Rešitev 2. naloge:

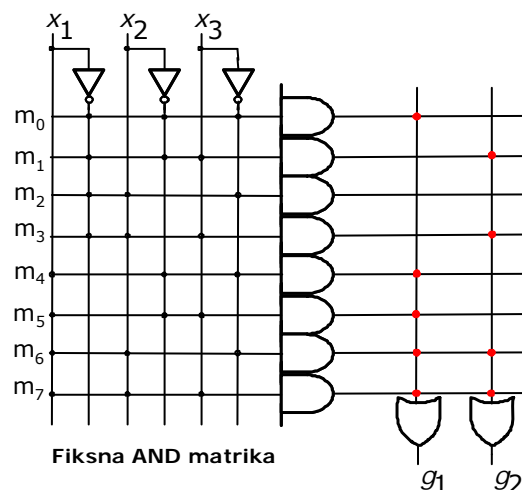
Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ev diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$\begin{aligned} g_1(x_1, x_2, x_3) &= x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= V(4, 6, 5, 7, 0) \end{aligned}$$

Podobno storimo še za preostale funkcije:

$$\begin{aligned} g_2 &= \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) \\ g_2(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\ g_2(x_1, x_2, x_3) &= V(1, 3, 6, 7) \end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ( $x_1 \ x_2 \ x_3$ ) od  $m_0$  do  $m_7$ . Številka minterma določa naslov lokacije ROM pomnilnika.



Narišemo celotno vezje ROM strukture in vstavimo pike (●) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

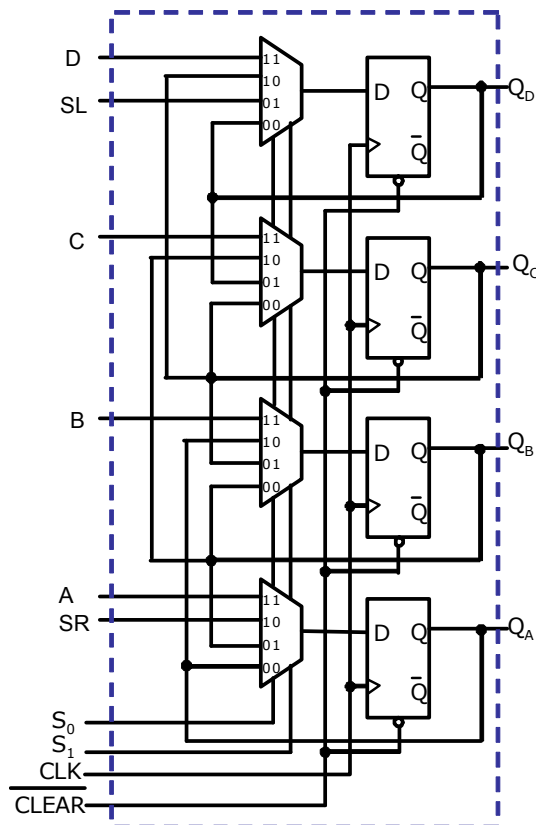
Vsako od operacij izpišemo v pravilnostno tabelo v kateri združimo funkcijska bita  $S_1$  in  $S_0$  in trenutno stanje na  $i$ -tem mestu registra  $Q_i(t)$ . Mesta registra od leve proti desni so  $Q_i = (Q_A, Q_B, Q_C, Q_D)$ . Realizacija z D flip-flopi nam analizo močno poenostavi, zaradi enačbe D flip-flopa:  $D = Q(t+1)$ .

Tabela 1: Prehajanje stanj univerzalnega registra.

$S_1$	$S_0$	$Q_i(t+1)$	funkcija
0	0	$Q_i(t)$	HOLD
0	1	$Q_{i+1}(t)$	SHR
1	0	$Q_{i-1}(t)$	SHL
1	1	$x_i$	LOAD

Iz poenostavljene tabele prehajanja stanj univerzalnega registra sestavimo realizacijo, ki bo vključevala izbiralnike MUX 4/1 in D-FF.

Na naslovna vhoda vseh MUX 4/1 vodimo funkcijska signala  $S_1$  in  $S_0$ . Potem na vsakem podatkovnem vhodu realiziramo ustrezno funkcijo.

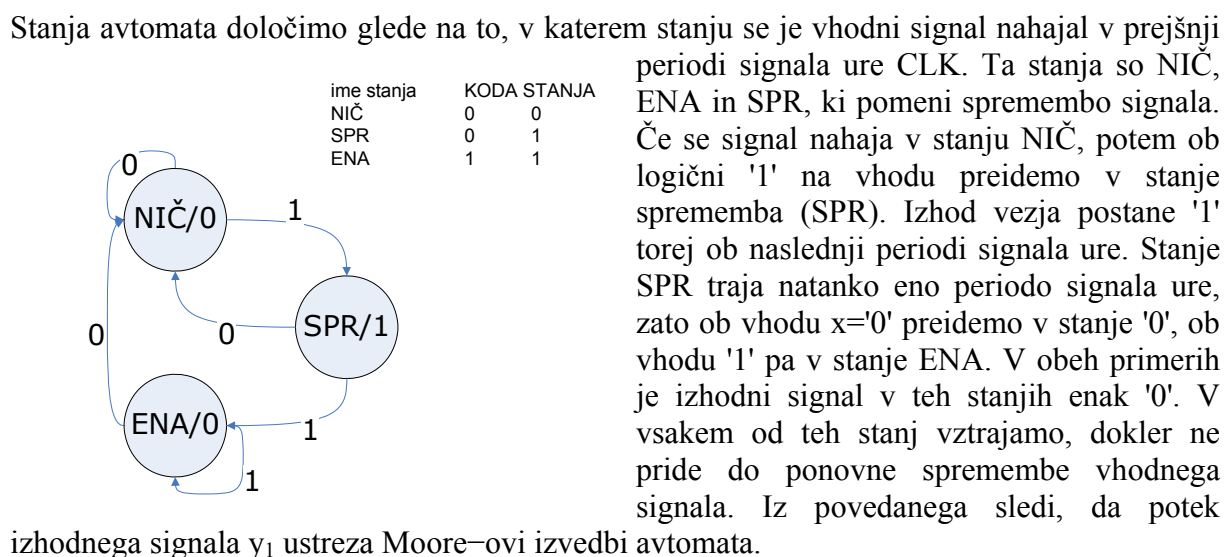


Stanje  $S_1S_0 = "00"$  pomeni držanje stanja (*HOLD*), torej bodo trenutne vrednosti D-FF ohranile vrednost  $Q_i(t+1) = Q_i(t)$ . Na sliki to realiziramo tako, da vodimo izhod D-FF nazaj na vhod pri podatkovnem vhodu 00. Stanje  $S_1S_0 = "01"$  pomeni pomik desno (*SHR* – ang. shift right), torej bodo D-FF pomaknili vsebino eno mesto desno. Pomik desno pomeni, da na mesto skrajno levega bita vpišemo vrednost zaporednega vhoda SR, nato  $Q_A$  vodimo na vhod  $Q_B$  in tako do skrajno desnega bita. Stanje  $S_1S_0 = "10"$  pomeni pomik levo (*SHL* – ang. shift left), torej bodo D-FF pomaknili vsebino eno mesto levo. Pomik levo pomeni, da na mesto skrajno desnega bita vpišemo vrednost zaporednega vhoda SL, nato  $Q_D$  vodimo na vhod  $Q_C$  in tako do skrajno levega bita.  $S_1S_0 = "11"$  pomeni vzporedno nalaganje z vhodov (*LOAD*)  $Q_D(t+1) = D$ ,  $Q_C(t+1) = C$ ,  $Q_B(t+1) = B$ ,  $Q_A(t+1) = A$ . Na tabeli smo  $i$ -ti vhod za vzporedno nalaganje označili kot  $x_i = (A, B, C, D)$ .

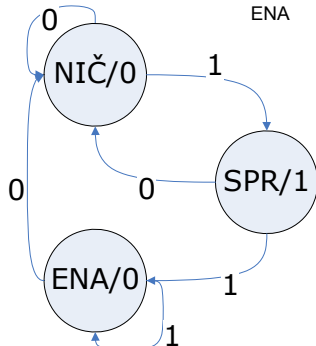
#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj.

Diagram stanj:



ime stanja	KODA STANJA
NIČ	0
SPR	1
ENA	1

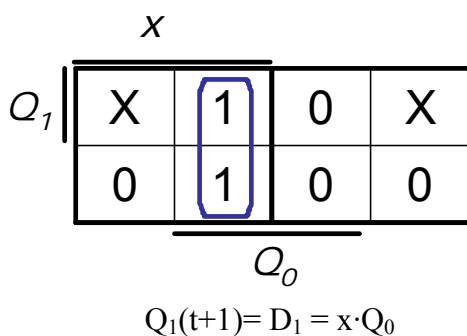


Narišemo tabelo prehajanja stanj

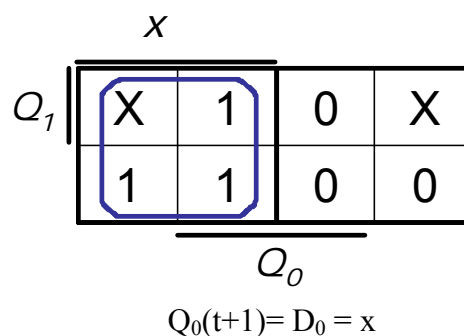
vhod in trenutno stanje			naslednje stanje		enačbe FF in izhoda		
x	$Q_1$	$Q_0$	$Q_1$	$Q_0$	$D_1$	$D_0$	y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	X	X	X	X	X
0	1	1	0	0	0	0	0
1	0	0	0	1	0	1	0
1	0	1	1	1	1	1	1
1	1	0	X	X	X	X	X
1	1	1	1	1	1	1	0

Iz tabele prehajanja stanj avtomata določimo enačbe D-FF:

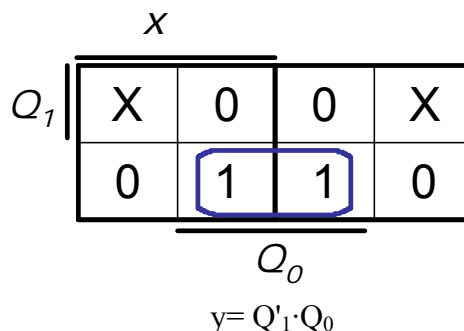
Za  $Q_1$  narišemo Veitchev diagram



Podobno za  $Q_0$  narišemo Veitchev diagram



In še za izhod y:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

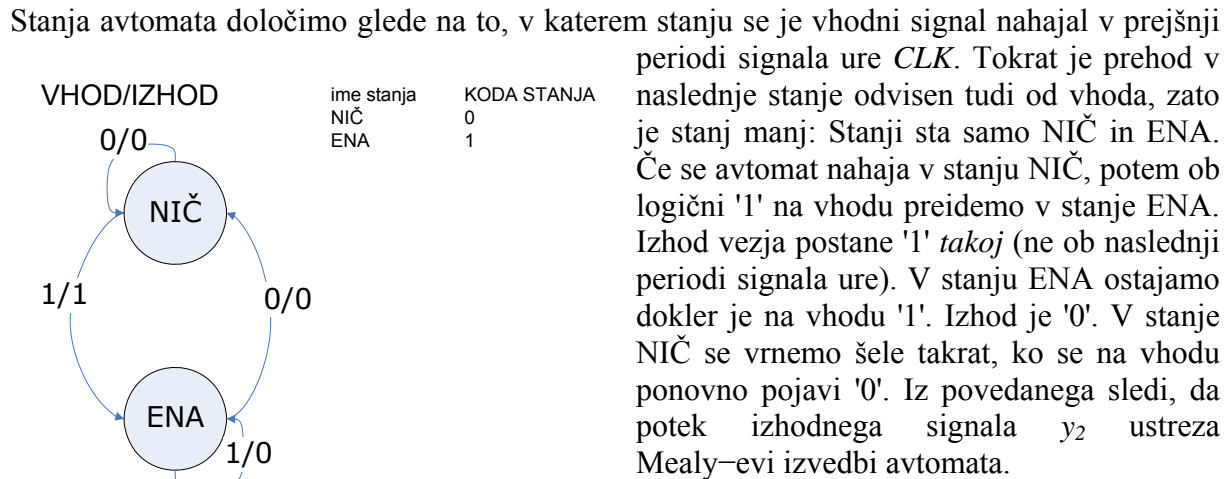
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Mealy–eva realizacija avtomata končnih stanj.

Diagram stanj:



Narišemo tabelo prehajanja stanj:

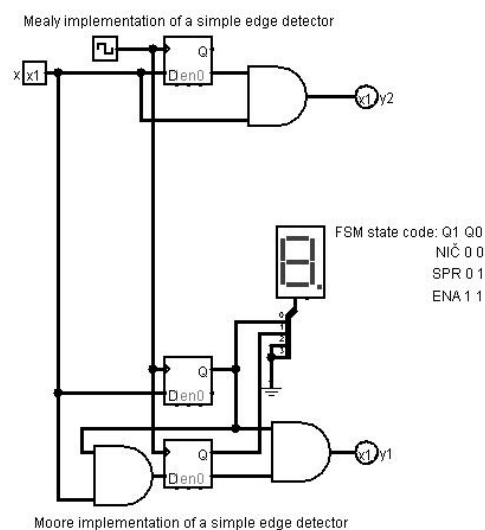
vhod in trenutno stanje		naslednje stanje	D–FF in izhod	
x	Q	Q	D	y
0	0	0	0	0
0	1	0	0	0
1	0	1	1	1
1	1	1	1	0

Veitchevih diagramov nam ni treba risati, saj enačbi D–FF in izhoda neposredno sledita iz tabele.

$$Q(t+1)=x \quad \text{in} \quad y=x \cdot Q'(t)$$

Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:

Logisim\fsm\front\_edge\_detector\_mealy\_moore.circ



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 09. 02. 2012

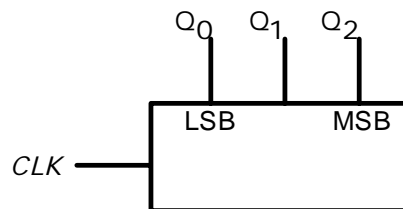
1. Določite redundance podane funkcije  $f$  tako, da bo nastala funkcija linearna in izračunajte koeficiente linearnosti.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

2. Realizirajte funkcijo  $f$  z dvovhodnimi vpoglednimi tabelami (ang. look-up table).

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

3. Prikažite sintezo sinhronnega 3-bitnega števca navzdol z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov ter vezje narišite. Imena signalov so razvidna iz spodnje slike.

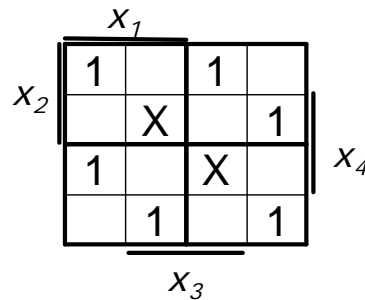


4. Narišite diagram stanj za avtomat končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z=1$ , ko se na vhodu pojavi zaporedje **110** ali **101**, sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda. Tip avtomata je razviden iz podanega časovnega zaporedja.

$CLK$	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
$w$	0	1	1	0	1	1	0	1	1	1	0	0	0
$z$	—	0	0	1	1	0	1	1	0	0	1	0	0

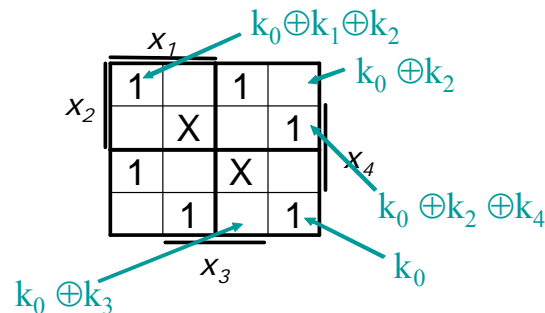
Rešitev 1. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:



Funkcija vsebuje same diagonalne člene, zato realizacija v obliki KNO oz. DNO ne nudi minimalne oblike. Če se izkaže, da je funkcija linearna, jo lahko realiziramo s pomočjo XOR funkcij. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

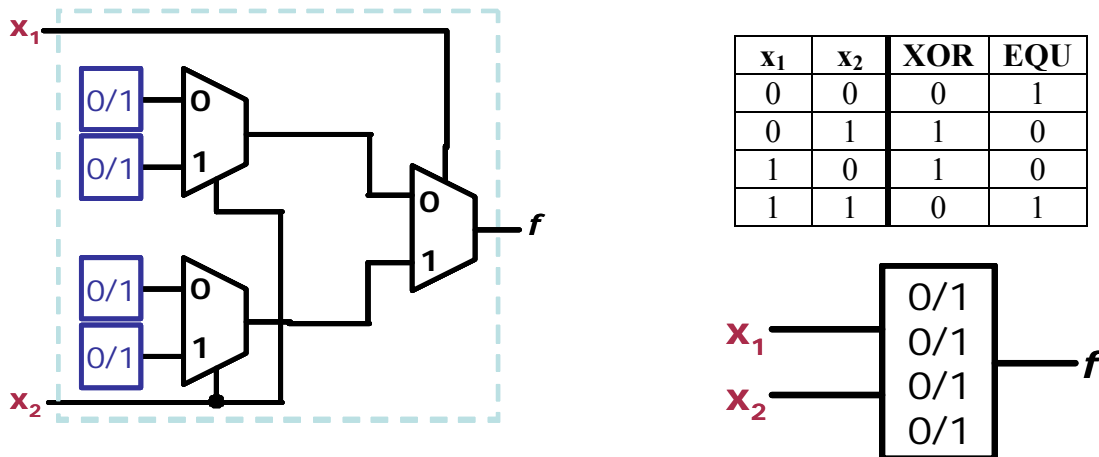
$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

## Rešitev 2. naloge:

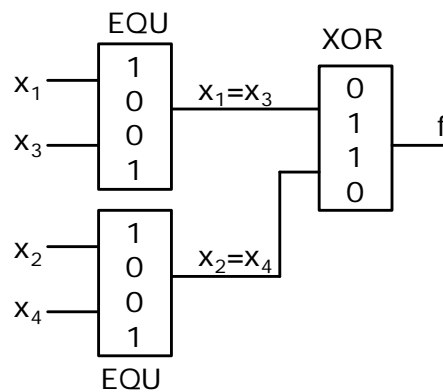
Funkcijo v je že primerna za realizacijo z dvovhodnimi vpoglednimi tabelami (ang. look-up table) v vezju FPGA in je ni treba posebej predelovati.

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

Dvovhodne vpogledne tabele (LUT2) so sestavljene iz pomnilnika (4 RAM celice) in treh 2/1 izbiralnikov. V vsako RAM celico so vpisane vrednosti, ki realizirajo eno od 16 osnovnih dvovhodnih funkcij.



Na zgornji sliki sta prikazani struktura dvovhodne vpogledne tabele (levo) in posplošeni simbol, ki ga uporabljamo pri risanju realizacij funkcij (desno) ter primer vsebine RAM celic za nekaj osnovnih funkcij (XOR, EQU). Tako lahko LUT uporabljamo za realizacijo funkcij v več nivojih.



Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za tri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števcov, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF:

Iz stolpca  $T_0$  se vidi:

$$T_0 = 1$$

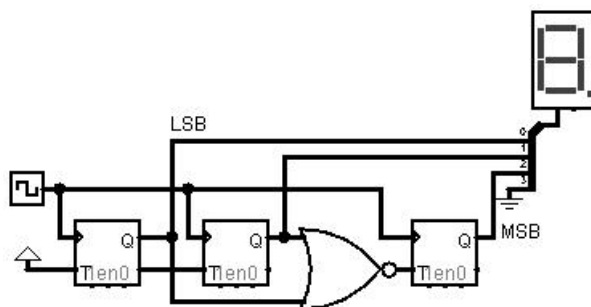
Z opazovanjem stolpcev trenutnega stanja določimo  $T_1$ :

$$T_1 = \overline{Q_0}$$

Podobno lahko določimo  $T_2$ :

$$T_2 = \overline{Q_0} \cdot \overline{Q_1} = \overline{Q_0 + Q_1}$$

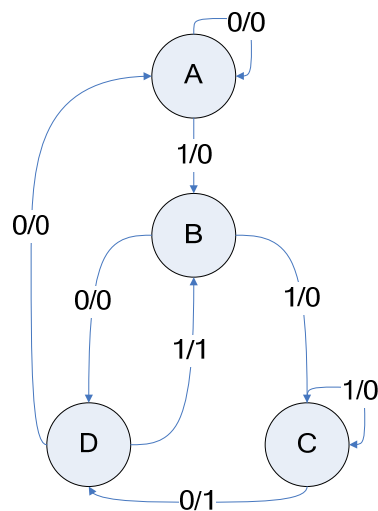
Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter 7 0 using T FF.circ:



#### Rešitev 4. naloge:

Naloga zahteva realizacijo z Mealy–evim tipom avtomata. Zapišemo začetno stanje A, v katerem ostajamo toliko časa, dokler se ne začne ena od sekvenc, ki ju zaznavamo. Obe sekvenci se začneta z '1', zato v stanje B preidemo, ko je na vhodu prva '1'. V stanju B ne moremo ostati, saj se na vhodu lahko pojavi '0' ali '1' – v obeh primerih gre za del zaznavanega zaporedja "10X" ali "11X". Iz stanja B preidemo v stanje C, če se vmes pojavi '1', tako da v tem stanju pomeni detekcijo sekvence "11X", v stanje D pa preidemo če se pojavi na vhodu '0', kar pomeni detekcijo sekvence "10X".

Prekrivanje zaporedij: Če se v stanju C pojavi '1' na vhodu, potem gre za sekvenco "111" na vhodu – kar še vedno pomeni, da ostajamo v stanju C, saj je prekrivanje vzorcev dovoljeno. Drugače se diagram obnaša, ko smo v stanju D in pride na vhod še ena '0' – takrat smo imeli na vhodu sekvenco "100", tako da se moramo vrniti v stanje A, saj se nobena od zaznavanih sekvenc ne začneja z '0'.



Delovanje avtomata preizkusimo na testnem zaporedju:

stanje	A	B	D	B	D	A	B	C	D	B	D	A	B	C	D	B	D	B	C	D	A	B	C	D
w	0	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	1	1	0	0	1	1	0
z	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	1

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 30. 01. 2013

1. Realizirajte funkcijo  $f$  v obliki PDNO z redundancami s čim manj izbiralniki 4/1.

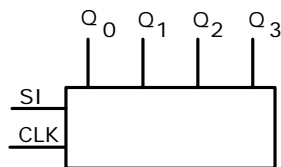
$$f(x_1, x_2, x_3, x_4) = V(1, 2, 9, 13, 15) \text{ in } V_x(0, 5, 11, 12)$$

2. Uporabite ROM vezje za realizacijo naslednjih funkcij:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2$$

ROM vezje ima 3 naslovne spremenljivke in 2 bitno vsebino. Narišite shemo elementa in v shemi označite programirane povezave oz. 'varovalke' s piko (●).

3. Narišite vezje 4-bitnega pomikalnega registra s T-celicami in logičnimi vrati. Register ima zaporedni vhod  $SI$  (ang. serial input) in vzporedni izhod ( $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ ).



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat (npr. tla prostora). Vezje ima 2-bitni izhod za enosmerni motor:

Koda izhoda		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če uporabnik pritisne gumb VRATA, se vrata začno pomikati navzgor. Če vrata na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če uporabnik pritisne gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

Podobno je v obratni smeri: Če vrata na poti naletijo na oviro ali pridejo do spodnje končne lege (tla prostora), se motor ustavi. Če uporabnik pritisne gumb VRATA, se ta začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

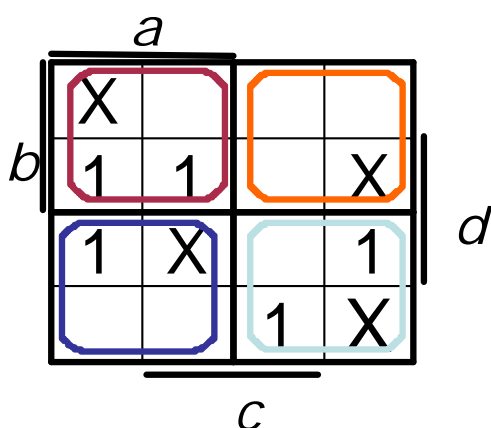
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Funkcija  $f$  je podana v obliki PDNO z redundancami.

$$f = V(1,2,9,13,15) \text{ in } V_x(0,5,11,12)$$

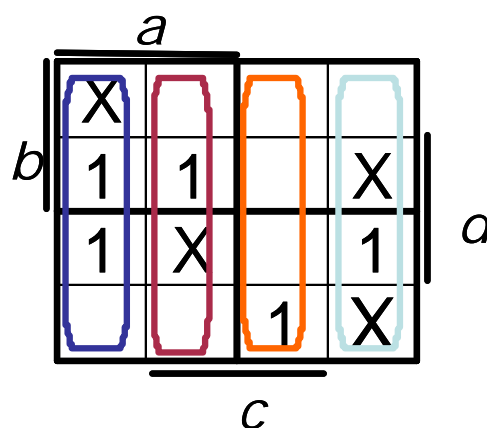
Dobljeno funkcijo vrišemo v Veitch-ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev-em diagramu. Če izberemo kot naslovni spremenljivki a b, potem dobimo:



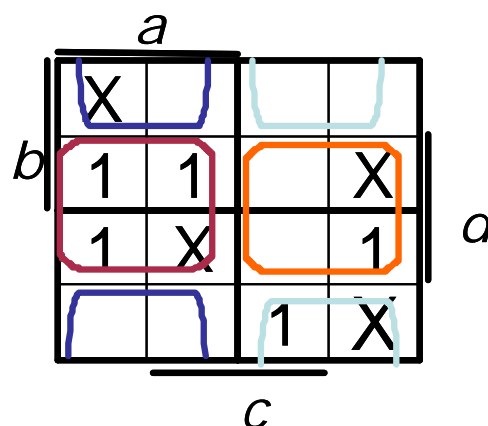
V zgornjem Veitch-evem diagramu so označena vsa štiri polja štirih mintermov, če izberemo vhodni spremenljivki a in b. Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $ab="11"$ , oranžni kvadrat ko bo  $ab="01"$ , temno modri ko bo  $ab="10"$  in svetlo modri ko bo  $ab="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata opišemo s spremenljivko d, če postavimo redundanco na '0'. Vrednost spodnjega desnega kvadrata je bolj komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $c \cdot d$ , za spodnjo '1' pa  $c \cdot d'$ . Funkcija bo torej  $c \cdot d + c \cdot d'$ , kar je enačba funkcije XOR. Najbolj enostavna realizacija je zgornji desni kvadrat, ki je kar '0', če postavimo redundanco na '0'. Zato, da bi pregledali še ostale možnosti, moramo

narisati še preostalih pet kombinacij dveh naslovnih vhodov.

Če izberemo kot naslovni spremenljivki a in c, dobimo levi Veitchev diagram, če a in d, pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najcenejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo inačice kvadratov, ki vsebujejo same '1' ali same '0'. Pri razvoju po a in c imamo pri  $ac="01"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'.



Pri razvoju po a in d nikjer ne nastopa ena sama '1' ali tri '1' ali diagonala (XOR) dveh '1'.



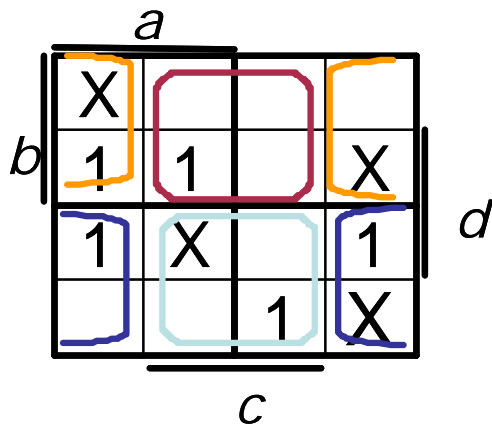
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

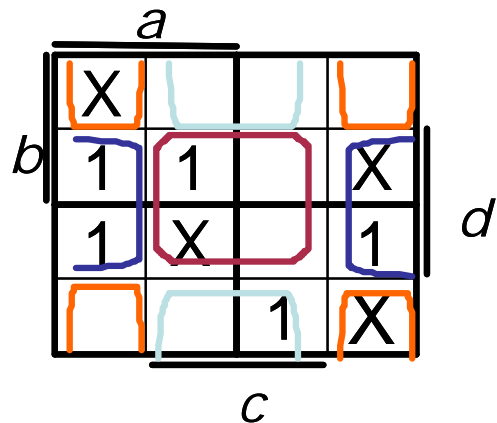
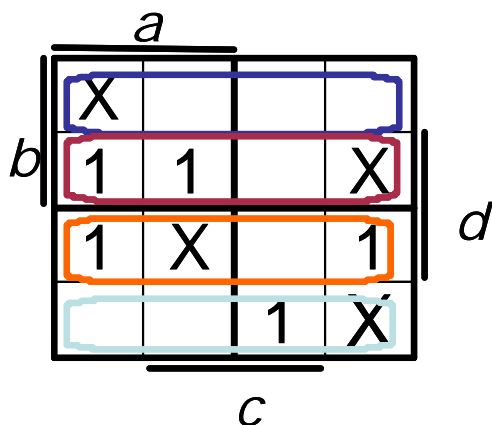
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Nato izberemo naslovni spremenljivki b in c, (levi Veitchev diagram) in b in d (desni diagram). Pri razvoju po b in c imamo pri bc="11" najneugodnejšo funkcijo (rdeč), saj vsebuje eno samo '1'.

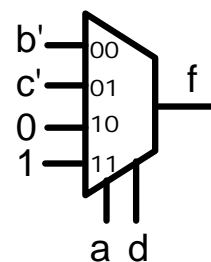


Pri razvoju po b in d pri bd="11" (rdeč) najneugodnejšo funkcijo, saj vsebuje tri '1'.



Zadnja kombinacija naslovnih vhodov je cd. Pri razvoju po c in d imamo pri cd="10" najneugodnejšo funkcijo (svetlo moder), saj vsebuje eno samo '1'. Najbolj ugodna kombinacija za realizacijo je torej razvoj po spremenljivkah a in d.

Končna realizacija funkcije:



## Rešitev 2. naloge:

Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ev diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$g_1(x_1, x_2, x_3) = x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3}$$

$$g_1(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3}$$

$$g_1(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$$

$$g_1(x_1, x_2, x_3) = V(4, 6, 5, 7, 0)$$

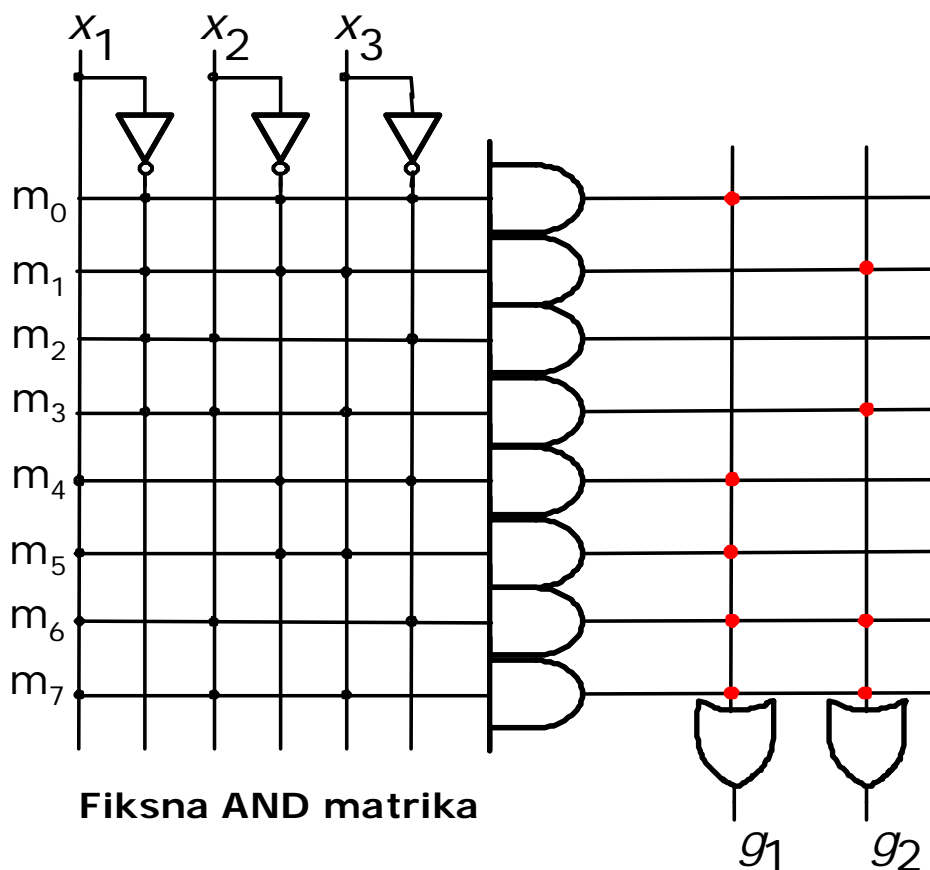
Podobno storimo še za funkcijo  $g_2$ :

$$g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3)$$

$$g_2(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3$$

$$g_2(x_1, x_2, x_3) = V(1, 3, 6, 7)$$

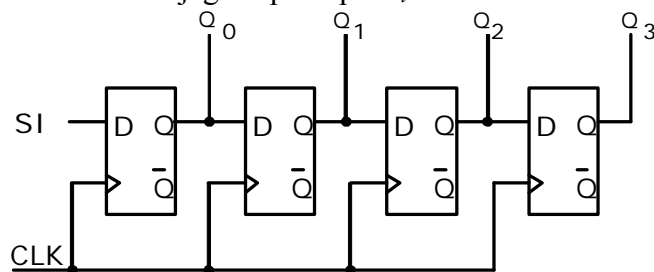
PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ( $x_1 x_2 x_3$ ) od  $m_0$  do  $m_7$ . Številka minterma določa naslov lokacije ROM pomnilnika.



Narišemo celotno vezje ROM strukture in vstavimo pike (•) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.

### Rešitev 3. naloge:

Zaporedno–vzporedni (SIPO) pomikalni register, realiziran s pomočjo D–FF, je veriga kaskadno vezanih D–FF, v kateri je izhod prejšnjega flip–flopa  $Q_{i-1}$  vezan na vhod naslednjega flip–flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz T–FF in logičnih vrat, moramo pravzaprav realizirati celico D–FF s pomočjo T–FF in logičnih vrat.

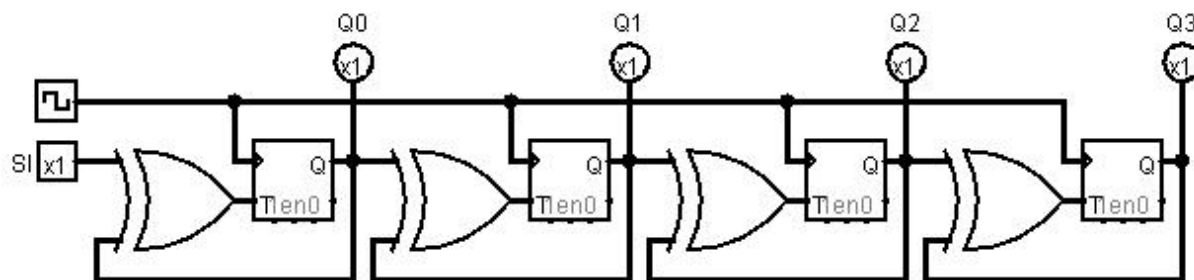
V ta namen zapišemo tabelo D–FF, pri kateri dodamo izhodni stolpec  $T$  vhoda.

$D$	$Q(t)$	$Q(t+1)$	$T$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Iz tabele sledi, da je  $T$  vhod XOR operacija  $Q(t)$  in vhoda D–FF, ki ga realiziramo.

$$Q(t+1) = Q(t) \oplus D$$

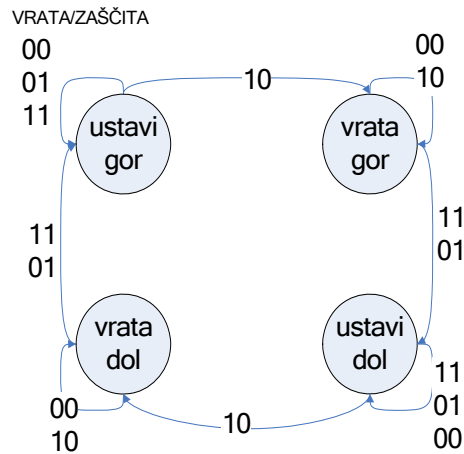
Če nastali D–FF sestavimo skupaj v 4–bitni pomikalni register dobimo spodnjo realizacijo.



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\shift\_reg\shift\_reg\_4bit\_using\_tff\_xor.circ

#### Rešitev 4. naloge:

Moore-ov diagram stanj avtomata krmilja za garažna vrata:



ime stanja	izhod stanja	
	OP <sub>1</sub>	OP <sub>0</sub>
ustavi gor	0	0
vrata gor	0	1
vrata dol	1	0
ustavi dol	0	0

Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si mora avtomat zapomniti v katero smer se bodo gibala vrata ob naslednjem pritisku na tipko VRATA.

Avtomat ima torej:

- stanje "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala navzgor,
- stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala navzdol.

Avtomat v stanju "ustavi gor" ostaja toliko časa, dokler uporabnik ne pritisne tipke (VRATA='1') in dokler na motorju ni napake (ZAŠČITA='0'), kar je kombinacija "10".

Vrata se pomikajo gor (avtomat preide v stanje "vrata gor"). V tem stanju lahko uporabnik tipko spusti (VRATA='0') in vrata se še vedno pomikajo navzgor. To se dogaja, dokler avtomat ne naleti na pogoj ZAŠČITA='1' (kombinaciji "11" in "01").

Ko postane ZAŠČITA='1', se avtomat postavi v stanje "ustavi dol" in v tem stanju ostaja dokler ostaja ZAŠČITA='1'. V tem stanju se motor ustavi in ZAŠČITA postane '0', ker čez motor ne teče tok. Če v tem stanju uporabnik pritisne tipko (VRATA='1') in je (ZAŠČITA='0') bo avtomat prešel v stanje "vrata dol". V tem stanju se vrata gibljejo navzdol, dokler ne naletijo na oviro (npr. tla prostora), ko preide v stanje "ustavi gor" ob pogoju (ZAŠČITA='1') ne glede na stanje tipke uporabnika.

# RAZVOJ DIGITALNIH SISTEMOV

Izpit  
12. 02. 2013

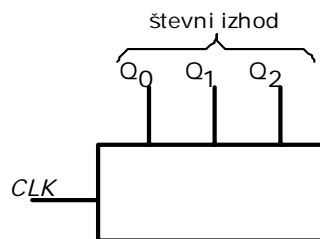
1. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi s čim manj izbiralniki 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 5, 7, 8, 9, 11, 12) \quad \text{in} \quad \&_x(3, 4, 10, 15)$$

2. Ali je funkcija  $f$  linearna? Če je linearna, potem izračunajte koeficiente linearnosti. Če ni linearna, potem utemeljite zakaj.

$$f^4 = V(0, 3, 4, 7, 9, 10, 13, 14)$$

3. Prikažite sintezo 3-bitnega sinhronnega števca navzdol po Graye-vi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2, Q_1, Q_0$ ). Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



Z uporabo D flip-flopov, ki so proženi na sprednji rob signala ure CLK, načrtajte Moore-ove avtomat končnih stanj, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov.

Krmilje ima:

- vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov,
- vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov,
- izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

Avtomat ne vrača drobiža in se ob detekciji plačila 15 centov ne vrača nazaj v začetno stanje, ampak ostane v končnem stanju. Vnos dveh kovancev naenkrat ni mogoč.

# Rešitev 1. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 5, 7 - 9, 11, 12) \text{ in } \&_x(3, 4, 10, 15)$$

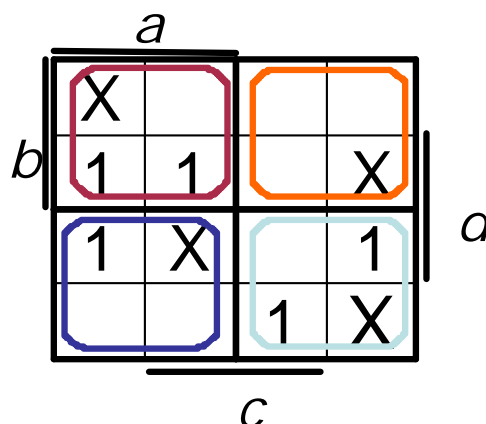
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f='0'$  za vse maksterme in  $f='X'$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f='1'$  in preberemo pri katerih mintermih je  $f='1'$  oz.  $f='X'$  ter funkcijo izrazimo v obliki PDNO.

Dobimo:

$$f = V(1, 2, 9, 13, 15) \text{ in } V_x(0, 5, 11, 12)$$

Dobljeno funkcijo vrišemo v Veitch-ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev-em diagramu. Če izberemo kot naslovni spremenljivki  $a$  b, potem dobimo:

$m$	$M$	$a$	$b$	$c$	$d$	$f$
0	15	0	0	0	0	X
1	14	0	0	0	1	1
2	13	0	0	1	0	1
3	12	0	0	1	1	0
4	11	0	1	0	0	0
5	10	0	1	0	1	X
6	9	0	1	1	0	0
7	8	0	1	1	1	0
8	7	1	0	0	0	0
9	6	1	0	0	1	1
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	X
13	2	1	1	0	1	1
14	1	1	1	1	0	0
15	0	1	1	1	1	1



V zgornjem Veitch-evem diagramu so označena vsa štiri polja štirih mintermov, če izberemo vhodni spremenljivki  $a$  in  $b$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $ab="11"$ , oranžni kvadrat ko bo  $ab="01"$ , temno modri ko bo  $ab="10"$  in svetlo modri ko bo  $ab="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata opišemo s spremenljivko  $d$ , če postavimo redundanco na '0'. Vrednost spodnjega desnega kvadrata je bolj komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $c \cdot d$ , za spodnjo '1' pa  $c \cdot d'$ . Funkcija bo torej  $c \cdot d + c \cdot d'$ , kar je enačba funkcije XOR. Najbolj enostavna realizacija je zgornji desni kvadrat, ki je kar '0', če postavimo redundanco na '0'. Zato, da bi pregledali še ostale možnosti, moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

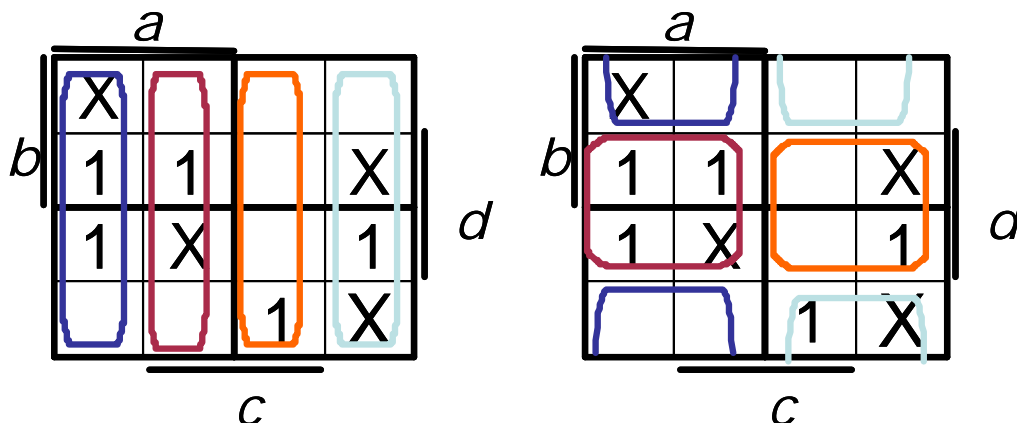
Če izberemo kot naslovni spremenljivki  $a$  in  $c$ , dobimo levi Veitchev diagram, če  $a$  in  $d$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najcenejšo realizacijo: Izogibamo se veliko različnim funkcijam

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

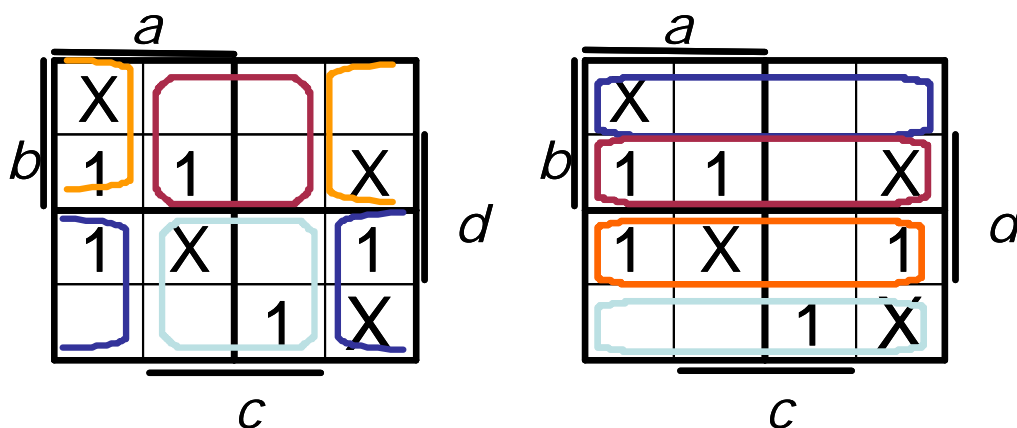
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

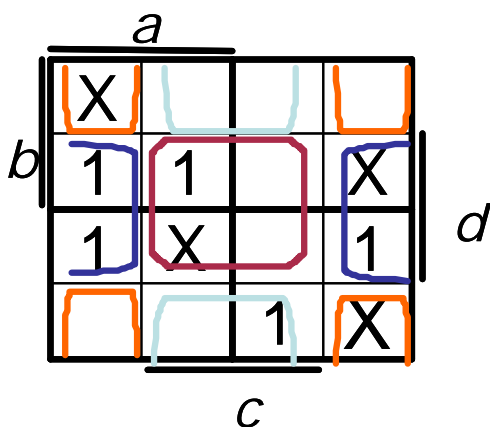
in iščemo inačice kvadratov, ki vsebujejo same '1' ali same '0'. Pri razvoju po a in c imamo pri ac="01" najneugodnejšo funkcijo, saj vsebuje eno samo '1'; medtem ko je razvoju po a in d nikjer ne nastopa ena sama '1' ali tri '1' ali diagonala (XOR) dveh '1'.



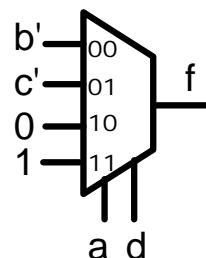
Nato izberemo naslovni spremenljivki b in c, (levi Veitchev diagram) in b in d (desni diagram). Pri razvoju po b in c imamo pri bc="11" najneugodnejšo funkcijo (rdeč), saj vsebuje eno samo '1'; medtem ko imamo pri razvoju po b in d pri bd="11" (rdeč) najneugodnejšo funkcijo, saj vsebuje tri '1'.



Zadnja kombinacija naslovnih vhodov je cd. Pri razvoju po c in d imamo pri cd="10" najneugodnejšo funkcijo (svetlo moder), saj vsebuje eno samo '1'. Najbolj ugodna kombinacija za realizacijo je torej razvoj po spremenljivkah a in d.



Končna realizacija funkcije:



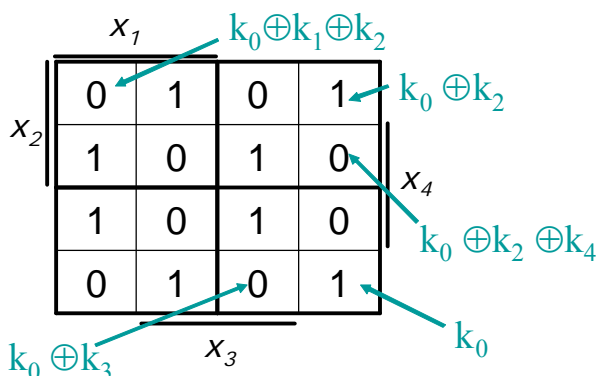
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 2. naloge:

Potrebno je določiti koeficiente linearnosti funkcije podane v PDNO. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat (recimo da  $x_4$  postane 1 v prvi iteraciji). Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Prepogibanje je prikazano v knjigi, stran 79, vaja 6.5.5.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4 \quad (1.1)$$

S pomočjo Veitch-evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $0 \oplus k_3=1 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=1$ , kar pomeni  $1 \oplus k_2=1$  sledi da je  $k_2=0$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=0$ , kar pomeni  $1 \oplus 0 \oplus k_4=0 \rightarrow k_4=1$ .

Če analiziramo naprej dobimo  $k_0 \oplus k_1 \oplus k_2=0$ , kar pomeni  $1 \oplus k_1 \oplus 0=0 \rightarrow k_1=1$ .

Če vstavimo dobljeno v enačbo (1.1) dobimo:  $k_0=1 \quad k_1=1 \quad k_2=0 \quad k_3=1 \quad k_4=1$

In rešitev:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= 1 \oplus x_1 \oplus x_3 \oplus x_4 \\ f(x_1, x_2, x_3, x_4) &= \overline{x_1 \oplus x_3 \oplus x_4} \\ f(x_1, x_2, x_3, x_4) &= (x_1 \equiv x_3 \equiv x_4) \end{aligned}$$



### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzdol po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
... 0, 4, 5, 7, 6, 2, 3, 1, 0, ...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
$Q_2$	$Q_1$	$Q_0$	$Q_2$	$Q_1$	$Q_0$	$T_2$	$T_1$	$T_0$
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1
0	1	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	0	1

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za  $T_0$  narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		$Q_2$			
$Q_1$		0	1	0	1
	$Q_0$	1	0	1	0

$$T_0 = Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za  $T_1$  narišemo Veitchev diagram

		$Q_2$			
$Q_1$		0	0	1	0
	$Q_0$	0	1	0	0

Za  $T_1$  sledi:

$$T_1 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_1 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

Operacija v oklepajih je XOR, zato enačbo lahko poenostavimo v:

$$T_1 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

In še za  $T_2$ :

		$Q_2$			
$Q_1$		1	0	0	0
	$Q_0$	0	0	0	1

Za  $T_2$  sledi:

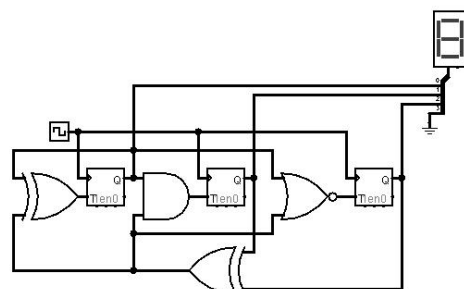
$$T_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot \overline{Q_1} + Q_2 \cdot Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR (za dve spremenljivki je to ekvivalenca), zato enačbo lahko poenostavimo v:

$$T_2 = (\overline{Q_2 \oplus Q_1}) \cdot \overline{Q_0} = \overline{(Q_2 \oplus Q_1)} \cdot \overline{Q_0}$$

T-FF down 3-bit Gray code counter: 0, 4, 5, 7, 6, 2, 3, 1, 0, ...



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta: Logisim\counter\3-bit down Gray code counter.circ

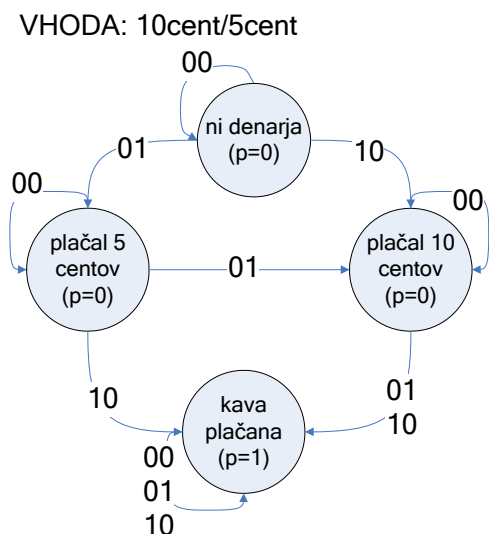
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj. Opis diagrama stanj:



Na začetku se nahajamo v stanju "ni denarja", v katerem je izhod  $p=0$ . Vhoda v avtomat sta dva: 10cent in 5cent, kar na diagramu kodiramo kot 10cent/5cent.

Mehanizem za vnos kovancev preprečuje hkraten vnos dveh kovancev, torej je kombinacija (10cent/5cent=11) nemogoča, zato bo avtomat od tu lahko prešel v poljubno stanje (X). Če uporabnik ni vrgel denarja v avtomat (10cent/5cent=00), potem ostaja v stanju "ni denarja". Če uporabnik vrže v avtomat 5 centov (10cent/5cent=01), potem preide v stanje "plačal 5 centov". Če uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "plačal 10 centov". Ne glede na to koliko je

vrgel bo izhod v teh dveh stanjih enak  $p=0$ , ker še ni plačal celotne cene kave. Če smo v stanju "plačal 5 centov" in uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ). Stanje "kava plačana" je končno in tam tudi ostanemo za vse možne kombinacije. Če smo v stanju "plačal 10 centov" in uporabnik vrže v avtomat 5 ali 10 centov (10cent/5cent=10 oz. 01), potem podobno preidemo v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ).

Naredimo tabelo prehajanja stanj:

trenutno stanje	10cent	5cent	naslednje stanje	izhod p
ni denarja	0	0	ni denarja	0
ni denarja	0	1	plačal 5 centov	0
ni denarja	1	0	plačal 10 centov	0
ni denarja	1	1	X	X
plačal 5 centov	0	0	plačal 5 centov	0
plačal 5 centov	0	1	plačal 10 centov	0
plačal 5 centov	1	0	kava plačana	0
plačal 5 centov	1	1	X	X
plačal 10 centov	0	0	plačal 10 centov	0
plačal 10 centov	0	1	kava plačana	0
plačal 10 centov	1	0	kava plačana	0
plačal 10 centov	1	1	X	X
kava plačana	0	0	kava plačana	1
kava plačana	0	1	kava plačana	1
kava plačana	1	0	kava plačana	1
kava plačana	1	1	X	X

Izberemo kodiranje stanj:

stanje	$Q_1$	$Q_0$
ni denarja	0	0
plačal 5 centov	0	1
plačal 10 centov	1	0
kava plačana	1	1

Nad tabelo prehajanja stanj uporabimo predlagano kodiranje stanj:

$Q_1$	$Q_0$	10cent	5cent	$Q_1$	$Q_0$	izhod p
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	X	X	X
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	X	X	X

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Naloga zahteva realizacijo z D–FF:

t				t+1				
Q <sub>1</sub>	Q <sub>0</sub>	10cent	5cent	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>1</sub>	D <sub>0</sub>	izhod p
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	1	X	X	X	X	X
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	1	1	0
0	1	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0	0
1	0	0	1	1	1	1	1	0
1	0	1	0	1	1	1	1	0
1	0	1	1	X	X	X	X	X
1	1	0	0	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	X	X	X	X	X

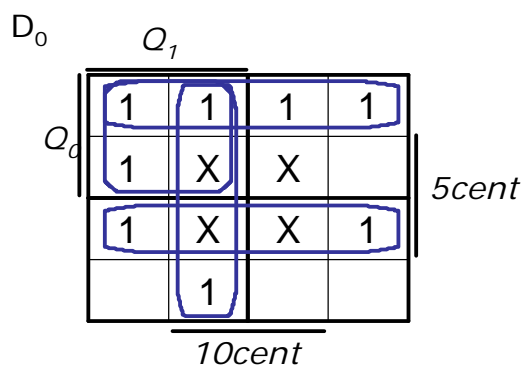
Iz dobljene tabele narišemo Veitch–eve diagrame za oba D–FF in izhod p:

$$D_0 = V(1, 4, 6, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$D_1 = V(2, 5, 6, 8, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$p = V(12-14) \text{ in } Vx(3, 7, 11, 15)$$

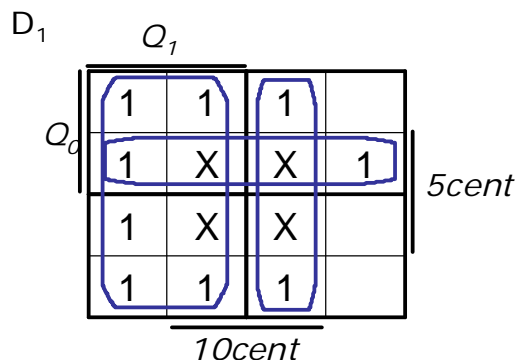
Veitch–ev diagram za D<sub>0</sub>:



$$D_0 = Q_1 \cdot Q_0 + \overline{5cent} \cdot Q_0 + 10cent \cdot Q_1 + 5cent \cdot \overline{Q_0}$$

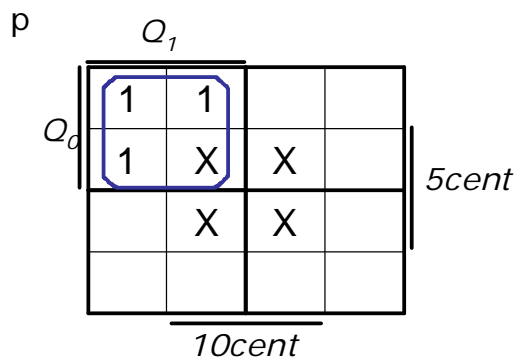
$$D_0 = Q_1 \cdot (Q_0 + 10cent) + 5cent \oplus Q_0$$

Veitch–ev diagram za D<sub>1</sub>:



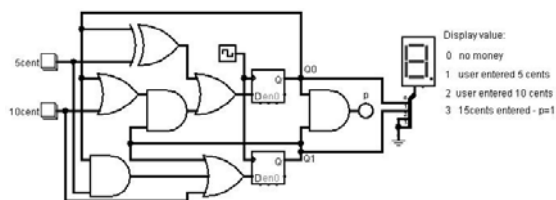
$$D_1 = Q_1 + 5cent \cdot Q_0 + 10cent$$

Veitch–ev diagram za izhod p:



$$p = Q_1 \cdot Q_0$$

Enačbo za izhod p bi lahko napisali tudi samo s sklepanjem, saj se izhod p postavi samo, ko je avtomat v stanju "kava plačana", ki ima kodo Q<sub>1</sub>Q<sub>0</sub>="11" – torej ko bosta Q<sub>1</sub> in Q<sub>0</sub> enaka '1', bo izhod p=1.



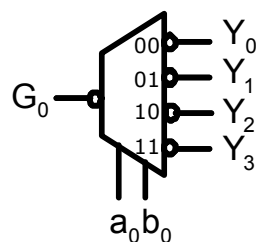
Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta: Logisim\fsm\Vending\_machine\_d\_ff\_5\_10\_cents\_price\_15cents.circ

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 19. 04. 2013

- Realizirajte funkcijo  $f^b = V(1, 2, 4, 7)$  z redundantnimi mintermi pri  $V_x(0, 3, 6)$  z enim TTL dekoderm 74139. Dekoder 74139 ima vhod za omogočenje elementa ( $G$ ) in izhode  $Y_0, Y_1, Y_2, Y_3$  v negativni logiki. Njegovo delovanje povzema spodnja tabela:

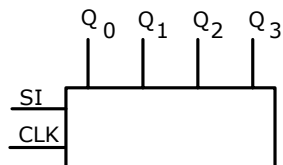
$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



- Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

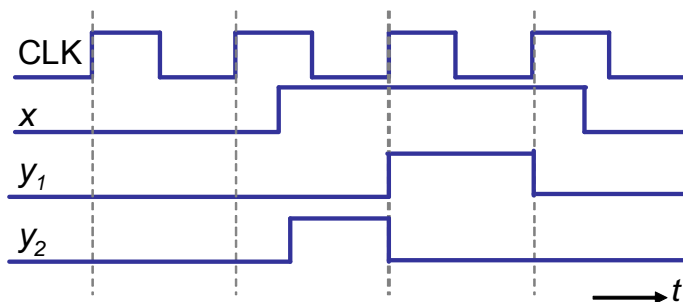
$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5-7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

- Narišite vezje 4-bitnega pomikalnega registra z JK celicami in logičnimi vrati. Register ima zaporedni vhod  $SI$  (ang. serial input) in vzporedni izhod ( $Q_0, Q_1, Q_2, Q_3$ ).



- Z uporabo D flip-flopov, ki so proženi na sprednji rob signala ure ( $CLK$ ) načrtajte Moore-ov **ali** Mealy-ev avtomat končnih stanj, katerega izhod  $y$  postane '1' ko se vhod  $x$  spremeni iz logične '0' na '1'.

Delovanje avtomata povzema spodnja slika. Narisani sta dve realizaciji izhoda avtomata  $y_1$  in  $y_2$ . Kateri izhod ( $y_1$  ali  $y_2$ ) pripada Mooreovemu tip avtomata in kateri Mealyevemu?



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

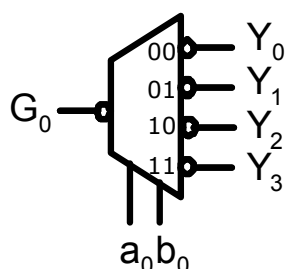
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Delovanje dekoderja 74139<sup>1</sup> povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Funkcijo  $f$  narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

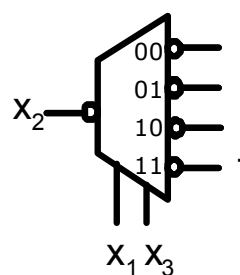
	$x_1$			
$x_2$	X	1	X	1
	1	0	1	X
	$x_3$			

Čim imamo na voljo dekodirnik z ENABLE vhodom v negativni logiki preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '1' vključno z redundancami X.

V zgornjem Veitch–evem diagramu je to spremenljivka  $x_2$ : Namreč, če je  $x_2=1$ , potem lahko vse redundance izberemo tako, da bo  $f=1$  za vse vrednosti  $x_2=1$ . Ko določimo spremenljivko za omogočenje elementa ( $G$ ), opazujemo samo preostali del Veitch–evega diagrama. Spodnje 4 vrednosti diagrama narišemo v novem Veitch–evem diagramu 2 spremenljivk.

	$x_1$	
$x_3$	0	1
	1	X

Dekoder ima aktivno nizke izhode, zato iz nastalega diagrama realiziramo *negacijo* funkcije  $f$  zato v Veitch–evem diagramu združujemo ničle, kar nastopa samo v primeru ko sta  $x_1=1$  in  $x_3=1$ .



<sup>1</sup><http://www.alldatasheet.com/view.jsp?Searchword=74HC139>

Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

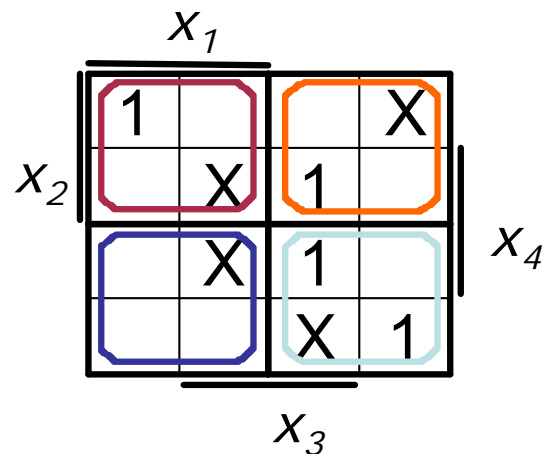
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundante maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

Dobljeno funkcijo vrišemo v Veitch–ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitch–ev diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:



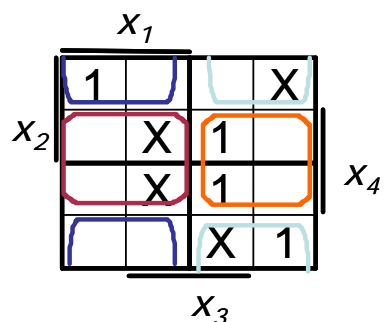
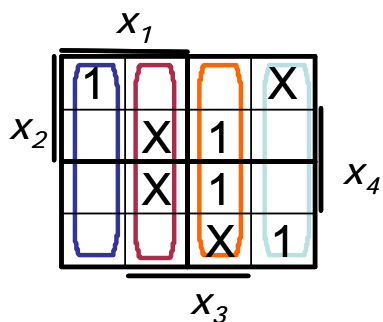
V zgornjem Veitch–evem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2="11"$ , oranžni kvadrat ko bo  $x_1x_2="01"$ , temno modri ko bo  $x_1x_2="10"$  in svetlo modri ko bo  $x_1x_2="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3x_4$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4$ . Funkcija bo torej  $x_3x_4 + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram, če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo inačice kvadratov, ki vsebujejo konstante (same '1' ali same '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .

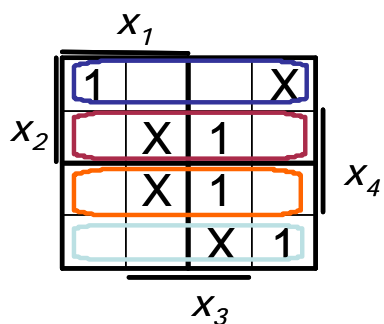
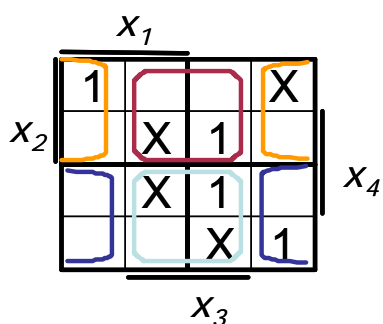
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

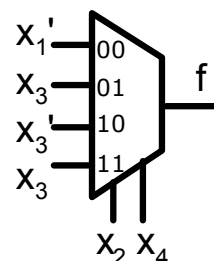
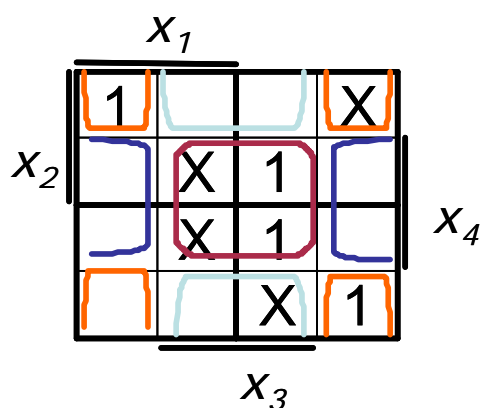
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Nato izberemo naslovni spremenljivki  $x_2$  in  $x_3$ , (levi diagram) in  $x_2$  in  $x_4$  (desni diagram). Pri razvoju po  $x_2$  in  $x_3$  imamo pri  $x_2x_3="00"$  najneugodnejšo funkcijo (moder), saj vsebuje eno samo '1'. Pri razvoju po  $x_2$  in  $x_4$  nikjer nimamo osamljene '1', zato se dajo funkcijski ostanki enostavno realizirati, če vse redundance postavimo na '1'.



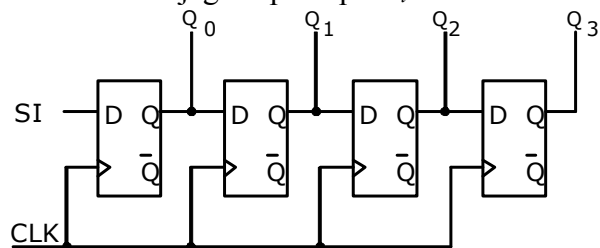
Zadnja kombinacija naslovnih vhodov je  $x_3$  in  $x_4$ . Pri razvoju po  $x_3$  in  $x_4$  imamo pri  $x_3x_4="00"$  najneugodnejšo funkcijo (oranžen), saj dve '1' opišemo s funkcijo ekvivalence. Možno rešitev torej predstavlja kombinacija naslovnih vhodov  $x_2$  in  $x_4$ .



Vezje izbiralnika je v predlogah avditornih vaj na domači strani predmeta:  
Logisim\MUX\mux\_4\_1\_f\_V\_0\_3\_7\_12\_in\_Vx\_2\_4\_11\_15.circ

### Rešitev 3. naloge:

Zaporedno-vzporedni (SIPO) pomikalni register, realiziran s pomočjo D-FF, je veriga kaskadno vezanih D-FF, v kateri je izhod prejšnjega flip-flopa  $Q_{i-1}$  vezan na vhod naslednjega flip-flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz JK-FF in logičnih vrat, je en način realizacije realizirati celico D-FF s pomočjo JK-FF in logičnih vrat.

V ta namen zapišemo tabelo D-FF, pri kateri dodamo izhodni stolpec JK vhodov.

$D$	$Q(t)$	$Q(t+1)$	$J$	$K$	Pomen stanja
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

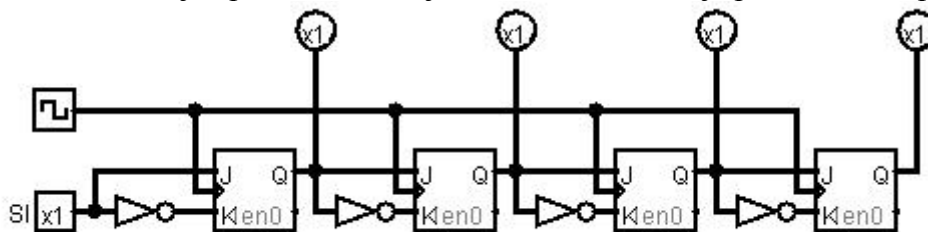
Iz tabele izrazimo  $J$  in  $K$  vhoda s pomočjo  $D$  vhoda in trenutnega stanja  $Q(t)$ . Če redundance (X) postavimo primerno, dobimo:

$$J = D$$

$$K = \overline{D}$$

Če nastali D-FF sestavimo skupaj v 4-bitni pomikalni register dobimo spodnjo realizacijo.

Ena možna rešitev je opisana realizacija D-FF z JK-FF, ki je prikazana na spodnji sliki.



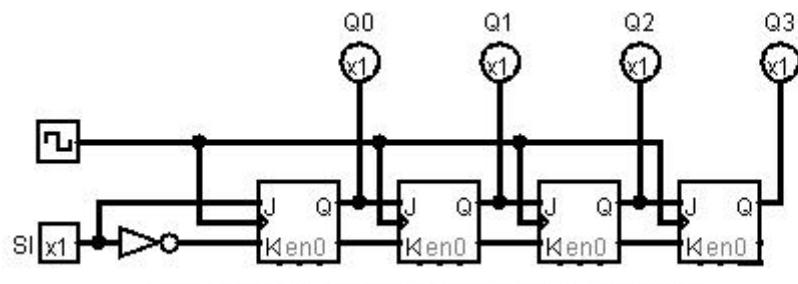
Druge možnost predstavlja neposredna realizacija s pomočjo tabele pomikanja med opazovanim mestom  $Q_i$  in naslednjim mestom  $Q_{i+1}$  pomikalnega registra. Ponovno zapišemo vzbujalno tabelo za opazovano mesto in določimo vhode glede na spremembo stanja ( $Q_{i+1}(t) \rightarrow Q_{i+1}(t+1)$ ) na  $(i+1)$  mestu.

Trenutna vsebina registra		Vsebinska registra ob pomiku	Vhoda FF na mestu $i+1$		Pomen stanja
$Q_i(t)$	$Q_{i+1}(t)$	$Q_{i+1}(t+1)$	$J_{i+1}$	$K_{i+1}$	
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

Če v dobljenih vseh za mesto  $(i+1)$  izberemo primerne redundance, dobimo vhoda  $J_{i+1} = Q_i(t)$  in  $K_{i+1} = Q_i(t)'$ . To velja tudi za vhodno mesto, torej bomo na vhodni JK-FF vezali vhod  $J_0 = x(t)$  in  $K_0 = x(t)'$ .



Neposredna realizacija pomikalnega registra z JK–FF je prikazana na spodnji sliki.



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\shift\_reg\shift\_reg\_4bit\_using\_jkff.circ

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

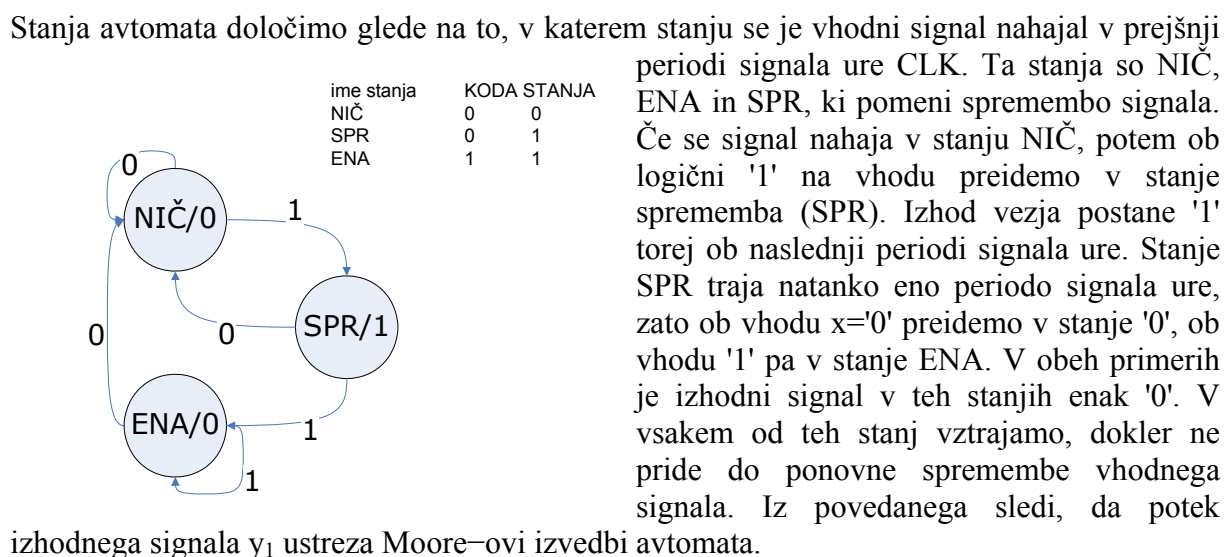
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

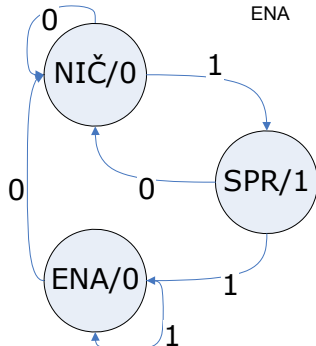
#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj.

Diagram stanj:



ime stanja	KODA STANJA
NIČ	0 0
SPR	0 1
ENA	1 1

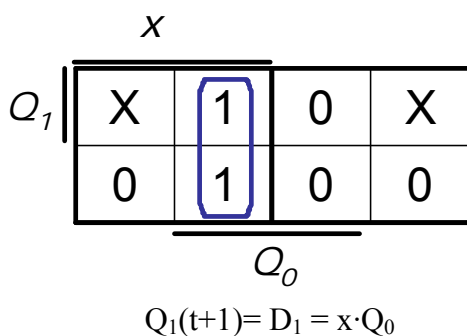


Narišemo tabelo prehajanja stanj

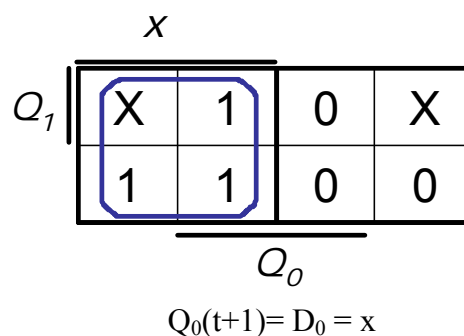
vhod in trenutno stanje			naslednje stanje		enačbe FF in izhoda		
x	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>1</sub>	D <sub>0</sub>	y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	X	X	X	X	X
0	1	1	0	0	0	0	0
1	0	0	0	1	0	1	0
1	0	1	1	1	1	1	1
1	1	0	X	X	X	X	X
1	1	1	1	1	1	1	0

Iz tabele prehajanja stanj avtomata določimo enačbe D-FF:

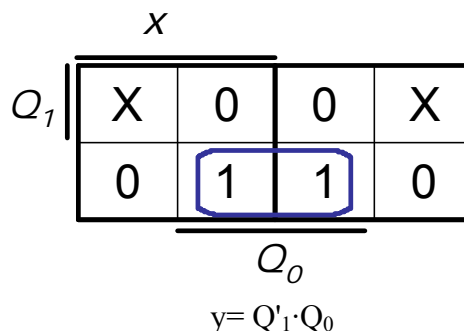
Za Q<sub>1</sub> narišemo Veitchev diagram



Podobno za Q<sub>0</sub> narišemo Veitchev diagram



In še za izhod y:



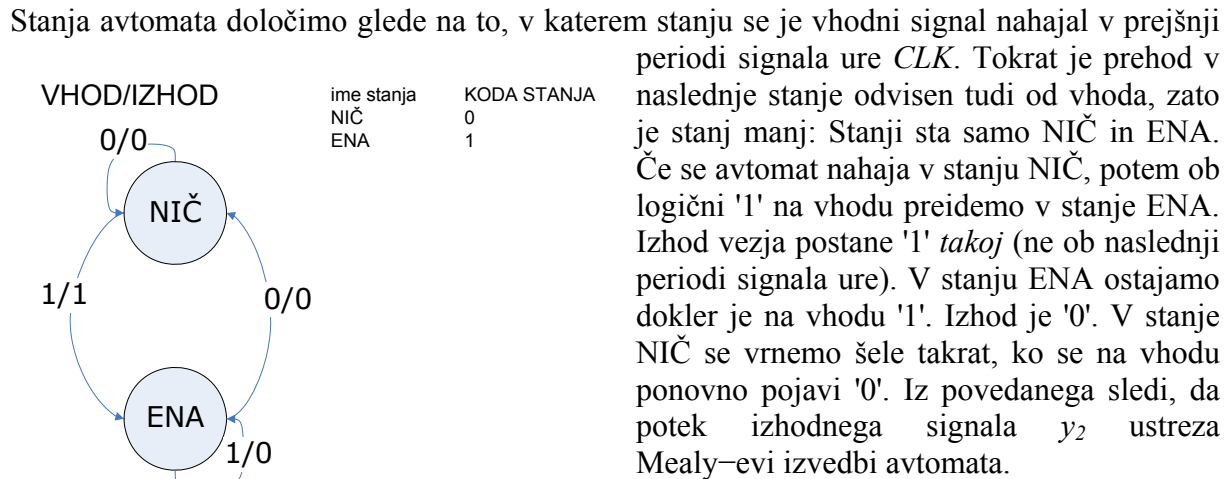
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Mealy–eva realizacija avtomata končnih stanj.

Diagram stanj:



Narišemo tabelo prehajanja stanj:

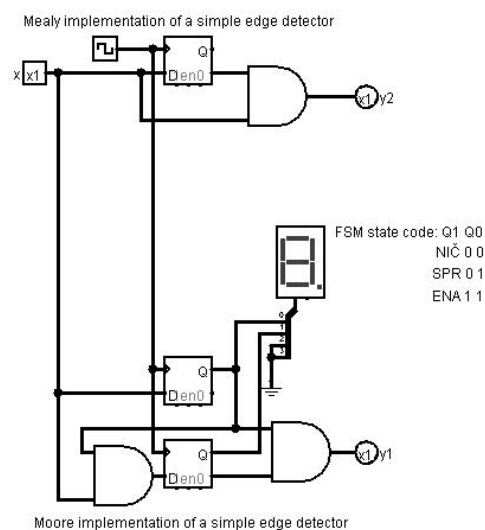
vhod in trenutno stanje		naslednje stanje	D–FF in izhod	
x	Q	Q	D	y
0	0	0	0	0
0	1	0	0	0
1	0	1	1	1
1	1	1	1	0

Veitchevih diagramov nam ni treba risati, saj enačbi D–FF in izhoda neposredno sledita iz tabele.

$$Q(t+1)=x \quad \text{in} \quad y=x \cdot Q'(t)$$

Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:

Logisim\fsm\front\_edge\_detector\_mealy\_moore.circ



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 19. 04. 2013

1. Realizirajte funkcijo  $f$  z dvovhodnimi vpoglednimi tabelami (ang. look-up table).

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

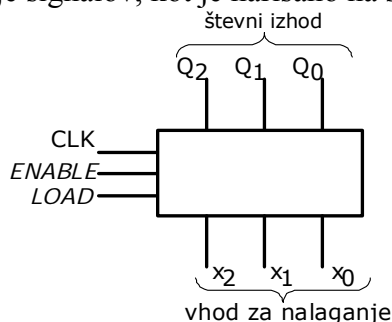
2. Realizirajte podano funkcijo  $f$  z redundancami s čim manj 4-bitnimi aritmetičnimi-logičnimi enotami (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Prikažite sintezo 3-bitnega sinhronega števca navzgor z omogočanjem štetja (*ENABLE*) in vzporednim nalaganjem (*LOAD*) z D flip-flopi, izbiralniki 2/1 in logičnimi vrati. Logika vseh krmilnih signalov je pozitivna.

S tovrstnim števcem realizirajte števec, ki šteje 2, 3, 4, 5, 2, 3, 4, 5 ...

Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Realizirajte Moore-ov avtomat končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z=1$  ko se na vhodu pojavi zaporedje "1001" ali "1111", sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno.

Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda.

w	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1
z	—	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

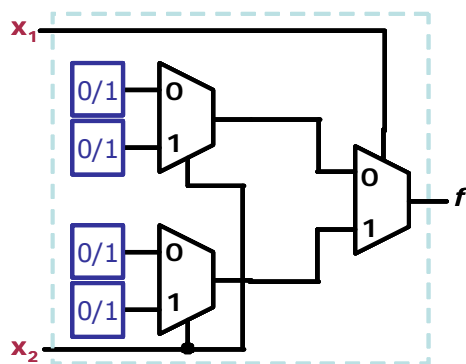
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

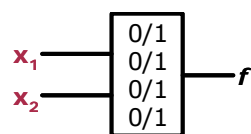
Funkcija je že primerna za realizacijo z dvovhodnimi vpoglednimi tabelami (ang. look-up table) v vezju FPGA in je ni treba posebej predelovati.

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

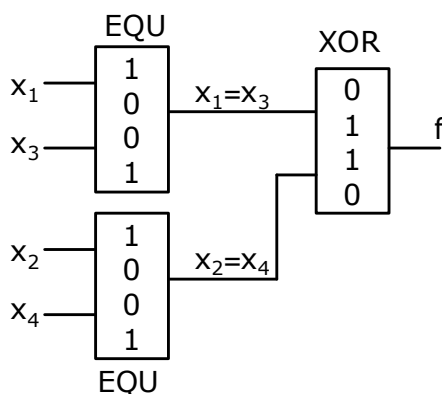
Dvovhodne vpogledne tabele (LUT2) so sestavljene iz pomnilnika (4 RAM celice) in treh 2/1 izbiralnikov. V vsako RAM celico so vpisane vrednosti, ki realizirajo eno od 16 osnovnih dvovhodnih funkcij.



$x_1$	$x_2$	AND	OR	XOR	EQU
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1



Na zgornji sliki sta prikazani struktura dvovhodne vpogledne tabele (levo) in posplošeni simbol, ki ga uporabljamo pri risanju realizacij funkcij (desno) ter primer vsebine RAM celic za nekaj osnovnih funkcij (XOR, EQU). Tako lahko LUT uporabljamo za realizacijo funkcij v več nivojih.



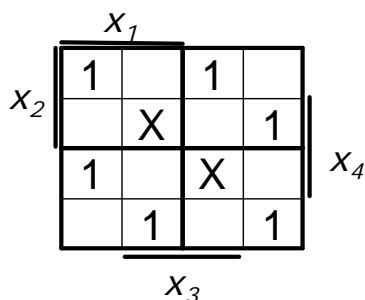
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:



Funkcija vsebuje same diagonalne člene, zato realizacija v obliki KNO oz. DNO ne nudi minimalne oblike. Če se izkaže, da je funkcija linearna, jo lahko realiziramo s pomočjo XOR funkcij. Linearnost funkcije ugotovljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

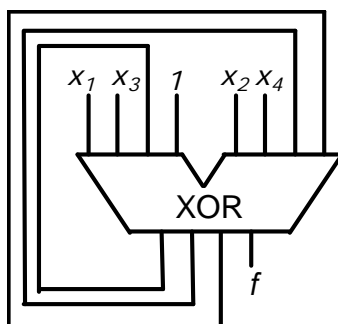
Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

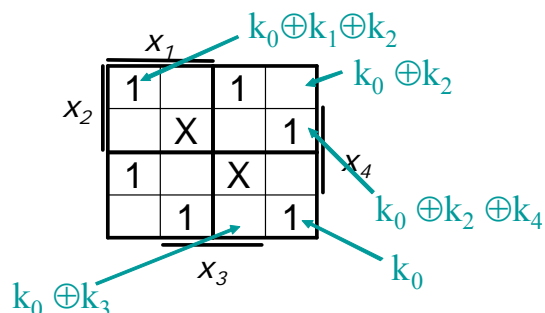
$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji uporabimo lastnost združevanja, ki velja za XOR funkcijo.

$$f(x_1, x_2, x_3, x_4) = 1 \oplus ((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$$



Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

### Rešitev 3. naloge:

trenutno stanje			naslednje stanje			D-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Podobno za D<sub>2</sub> narišemo Veitchev diagram

$D_2$ :

	$Q_2$			
$Q_1$	1	0	1	0
	1	1	0	0
	$Q_0$			

Za D<sub>2</sub> sledi:

$$D_2 = Q_2 Q_1' + Q_2 Q_0' + Q_2' Q_1 Q_0$$

Iz tabele prehajanja stanj števca določimo enačbe D-FF:

Za D<sub>0</sub> se iz tabele vidi  $D_0 = Q_0'$

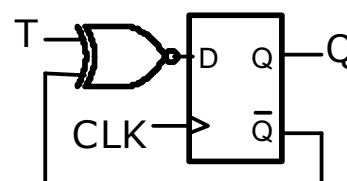
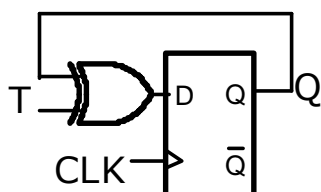
Za D<sub>1</sub> narišemo Veitchev diagram:

$D_1$ :

	$Q_2$			
$Q_1$	1	0	0	1
	0	1	1	0
	$Q_0$			

$$D_1 = Q_0 \oplus Q_1$$

Če enačbo  $D_0 = Q_0'$  zapišemo kot  $D_0 = 1 \oplus Q_0$  in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:



Slika 1: Realizacija T-FF s pomočjo D-FF in XOR vrat (levo) in XNOR vrat (desno)

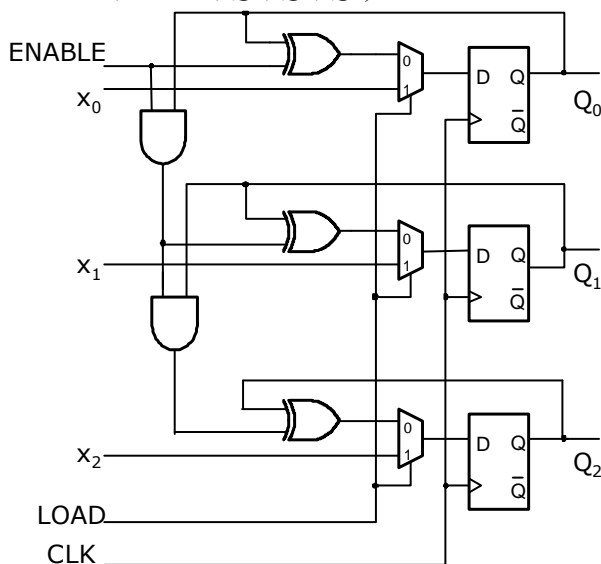
Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (*ENABLE*): Če prvemu "T-FF" (D-FF z XOR vrati) postavimo vhod  $T_0 = '0'$  namesto  $T_0 = '1'$ , vsi FF ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod  $T_0$  postavimo zunanji signal *ENABLE*, števec ne bo štel, ampak ohranjal stanje, če bo *ENABLE* = '0'. V verigi sinhronnega števca so namreč vsi FF vezani tako, da so odvisni od prvega FF. Če stanje ohranja prvi, ga bodo tudi vsi ostali.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnega vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame za 5 spremenljivk (*ENABLE*, *LOAD*,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



**Slika 2: Sinhroni števec z vzporednim nalaganjem (*LOAD*) in omogočanjem štetja (*ENABLE*) (3-bitna izvedba).**

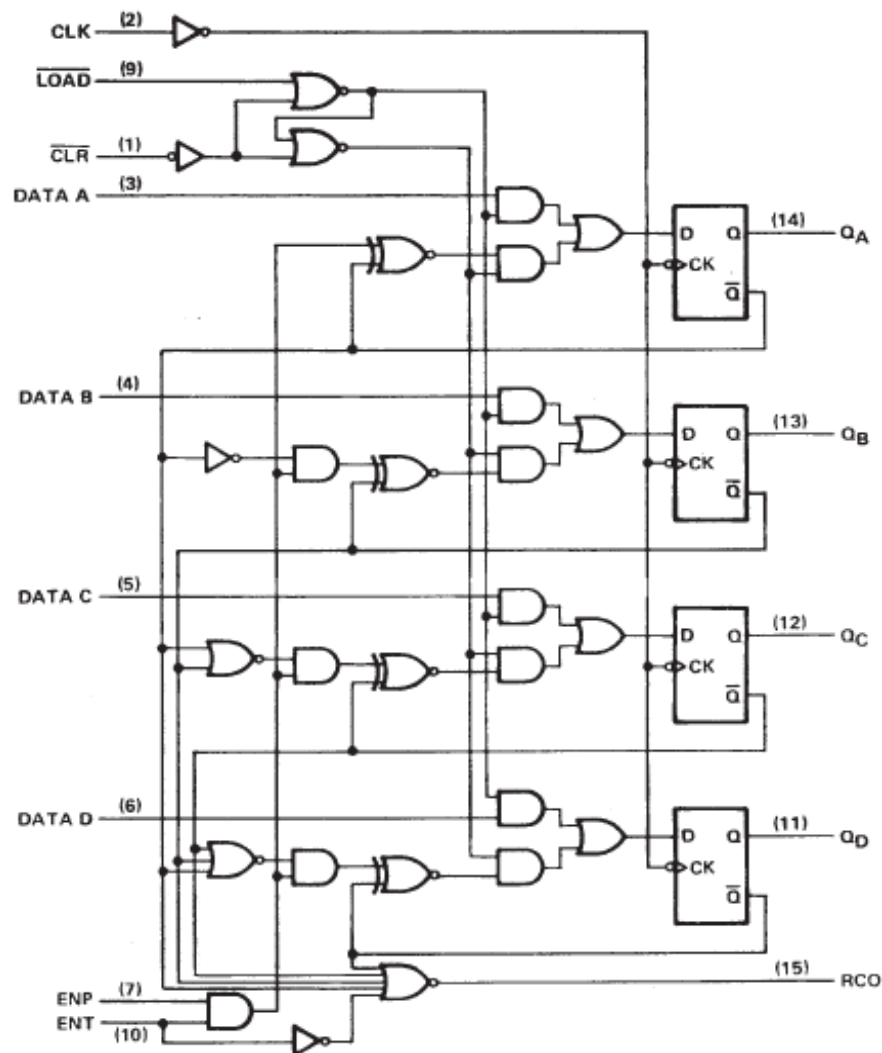
Če želimo z nastalim števcem šteti naraščajoče 2, 3, 4, 5 ..., moramo ko števec pride do stanja 5 ( $Q_2Q_1Q_0=101_2$ ) števec postaviti nazaj na stanje v stanje 2 ( $Q_2Q_1Q_0=x_2x_1x_0=010_2$ ), torej signal *LOAD* dekodiramo s pomočjo 2–vhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se  $Q_2 = '1'$  in  $Q_0 = '1'$  v števeni sekvenci pojavlja samo enkrat – če bi se večkrat, bi morali dekodirati tudi  $Q_1$ . Pri tovrstnih števcih ponavadi uporabljamo še en signal *RESET*, s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod D–FF za brisanje asinhrono priključimo signal *RESET*.

Nastali strukturi števca bi lahko na isti način dodali še četrti bit. Tako bi dobili podobno strukturo kot je 4–bitni sinhroni števec z vzporednim nalaganjem 74163<sup>1</sup>. Pri spodnji realizaciji tega vezja so uporabljeni D–FF, proženi na negativni rob signala ure (*CLK*). Štetje omogočimo s signaloma *ENP*, *ENT* (ang. enable parallel, enable transfer). Štetje je izvedeno tako, da se D–FF spremenijo v T–FF, kar dosežemo s pomočjo XNOR vrat, ki imajo en vhod vezan na števeni signal ( $ENT=ENP='1'$ ), drug vhod pa na izhod  $Q'$  FF. AND vrata pred XNOR zagotavljajo prenehanje štetja, če velja  $ENT \cdot ENP \neq '1'$ . Na vhodu D–FF je vezan enostaven izbiralnik iz dveh AND vrat, vezanih na OR vrata. Ta izbiralnik določa, ali bo števec štel, ali bo vzporedno naložil vrednost. Zgornja AND vrata izbiralnika so krmiljena s signalom  $LOAD \cdot CLR'$ , spodnja pa z  $LOAD' \cdot CLR'$ . Čim velja, da je  $CLR' = '1'$  in  $LOAD' = '0'$ , se v D–FF asinhrono naloži vsebina na vseh  $Q_D Q_C Q_B Q_A = DATA_D DATA_C DATA_B DATA_A$ , medtem ko dokler velja  $LOAD' = '1'$  in  $CLR' = '1'$ , bo števec štel navzgor. Če je  $CLR' = '0'$ , je na obeh vseh AND vrat izbiralnika '0' in stanje vseh D–FF se asinhrono postavi na  $Q_D Q_C Q_B Q_A = "0000"$ . Števeni izhodi so  $Q_D Q_C Q_B Q_A$ .

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74163>



Izhod *RCO* (*ripple carry out*) se postavi na '1', ko je števec v stanju "1111" ob tem, da je  $ENT=1$ . Signal *RCO* je izveden z NOR vrati na vhode katerih so priključeni negirani izhodi D-FF, kar je po De Morgan-ovem teoremu ekvivalent AND vratom.



Slika 3: Struktura 4-bitnega MSI sinhronega števca z vzporednim nalaganjem (74163).

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Prikazali bomo izvedbo avtomata končnih stanj Moore-ove izvedbe.

Diagram prehajanja stanj:

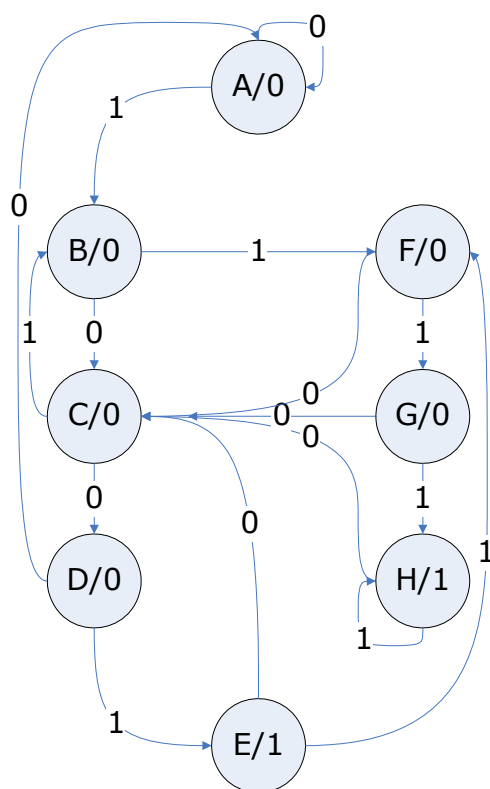


Diagram stanj začnemo risati tako, da najprej narišemo začetno stanje (A) v katerega se vračamo vedno, kadar sekvenca ne bo podobna tisti, ki jo zaznavamo (1111 oz. 1001). V tem stanju vztrajamo toliko časa, dokler je na vhodu '0', saj se nobena od sekvenc ne začne z '0'. Ko pride na vhod prva '1', preidemo v drugo stanje (B), v katerem imamo dve možnosti, saj se sekvenci razlikujeta na drugem mestu: Če na vhod tega stanja pride '0', bomo zaznavali sekvence tipa '10XX', če pa pride '1', potem pa sekvence tipa '11XX'. Recimo, da v stanju B na vhod pride '0', torej napredujemo v stanje C. V tem stanju se ponovno lahko pojavi '0' – torej bi bila na vhodu sekvenca

tipa '100X', kar bi nas vodilo do naslednjega stanja D. Če pa se pojavi na vhodu logična '1' je sicer sekvenca napačna, vendar moramo to predstaviti tako kot da je na vhod prišla že prva '1' od sekvence - naloga namreč pravi, da se sekvence lahko med seboj prekrivajo. S podobnim načinom razmišljanja narišemo naslednje stanje E, v katero napredujemo iz D samo, ko vanj pride '1', kar pomeni da smo v tem stanju zaznali sekvenco "1001", zato je v tem stanju izhod vezja enak '1'. Stanja F, G in H služijo pomnjenju koliko enic je prišlo na vhod vezja in sicer: stanje F pomenita "11" na vhodu vezja, stanje G tri enice in zadnje stanje H četrto enico sekvence. Slednje stanje vztraja, dokler je na vhodu '1', saj tako rešimo prekrivanje vzorca '1111'. Če pa v kateremkoli od stanj F, G in H pride na vhod '0', se postavimo v stanje C, saj to stanje pomeni, da je na vhodu sekvenca '10XX'.

Tabela prehajanja stanj:

trenutno stanje		naslednje stanje		izhod
pomen stanja	koda stanja	w=0	w=1	z
začetno stanje	A	A	B	0
prva enica '1001' ali '1111' na vhodu	B	C	F	0
prva ničla '1001' na vhodu	C	D	B	0
druga ničla '1001' na vhodu	D	A	E	0
druga enica '1001' na vhodu	E	C	F	1
druga enica '1111' na vhodu	F	C	G	0
tretja enica '1111' na vhodu	G	C	H	0
četrtta enica '1111' na vhodu	H	C	H	1

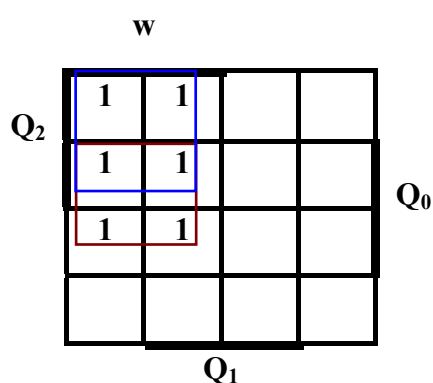
Za izvedbo bomo rabili najmanj 3 FF, saj je stanj 8. Izberemo zaporedno kodiranje stanj se pravi A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.

Vzbujalna tabela:

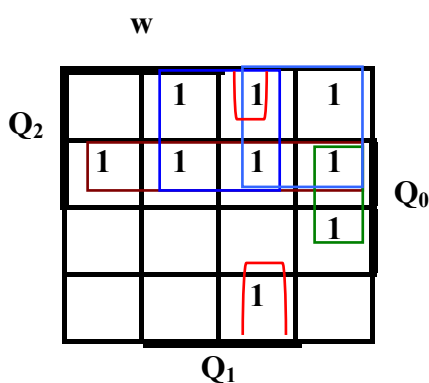
trenutno stanje				naslednje stanje						izhod
				w=0			w=1			
	Q <sub>2</sub> (t)	Q <sub>1</sub> (t)	Q <sub>0</sub> (t)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	z
A	0	0	0	0	0	0	0	0	1	0
B	0	0	1	0	1	0	1	0	1	0
C	0	1	0	0	1	1	1	0	1	0
D	0	1	1	0	0	0	1	0	0	0
E	1	0	0	0	1	0	1	0	1	1
F	1	0	1	0	1	0	1	1	0	0
G	1	1	0	0	1	0	1	1	1	0
H	1	1	1	0	1	0	1	1	1	1

Iz vzbujalne tabele sestavimo tri Veitcheve diagrame:

$Q_2(t+1)$ :



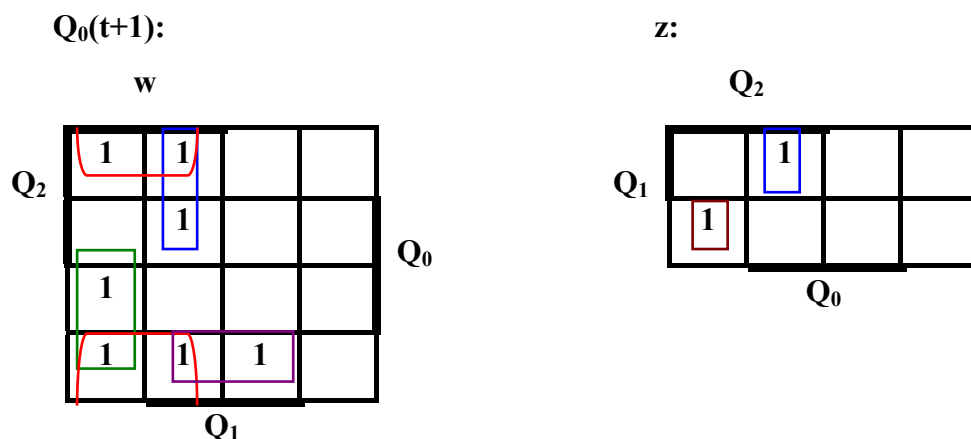
$Q_1(t+1)$ :



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Funkcije v Veitchevih diagramih minimiziramo in dobimo enačbe za realizacijo s pomočjo D flip-flopov. Realizacija s pomočjo D flip-flopov je najbolj enostavna, saj je enačba D flip-flopa:

$$D = Q(t+1) \quad (4.1)$$

kar pomeni, da lahko iz minimizacije Veitchevih diagramov naslednjih stanj zapišemo enačbe za vhode D flip-flopov:

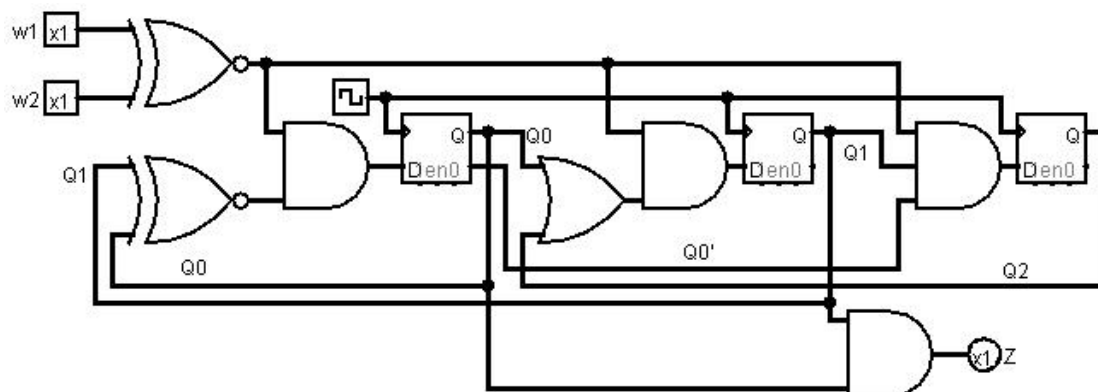
$$\begin{aligned} D_2 &= Q_2(t+1) = w \cdot Q_2(t) + w \cdot Q_0(t) = w \cdot (Q_2(t) + Q_0(t)) \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + \bar{Q}_1(t) \cdot Q_0(t) + Q_1(t) \cdot \bar{Q}_0(t)) = \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + Q_1(t) \oplus Q_0(t)) \\ D_0 &= Q_0(t+1) = w \cdot \bar{Q}_0(t) + w \cdot \bar{Q}_2(t) \cdot \bar{Q}_1(t) + w \cdot Q_2(t) \cdot Q_1(t) + \bar{Q}_1(t) \cdot \bar{Q}_0(t) = \\ D_0 &= Q_0(t+1) = w \cdot (\bar{Q}_0(t) + \bar{Q}_2(t) \oplus \bar{Q}_1(t)) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t) \end{aligned} \quad (4.2)$$

Izhod z lahko zapišemo kot:

$$z = Q_2(t) \cdot (Q_1(t) \cdot Q_0(t) + \bar{Q}_1(t) \cdot \bar{Q}_0(t)) = Q_2(t) \cdot (Q_1(t) \oplus Q_0(t)) \quad (4.3)$$

Vezje avtomata narišemo iz enačb (4.2) in (4.3).

Opis delovanja in vezje avtomata, ki primerja enakost znotraj 4 period signala ure je v predlogah vaj na domači strani predmeta v imeniku Logisim\fsm\fsm\_four\_periodes\_of\_equality.circ



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

11. 09. 2013

1. Realizirajte izbiralnik vodil (ang. bus multiplexer) s petimi 4-bitnimi vhodnimi vodili A, B, C, D, E. Glede na naslovni vhod  $a_2a_1a_0$  je izhod  $y$  določen s tabelo:

$a_2$	$a_1$	$a_0$	$y$
0	0	0	A
0	0	1	B
0	1	0	A
0	1	1	C
1	0	0	A
1	0	1	D
1	1	0	A
1	1	1	E

Uporabite lahko največ 3 od sledečih integriranih vezij:

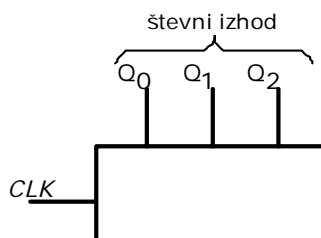
74153 (2\*MUX 4/1, skupni naslov)

74157 (4\*MUX 2/1, skupni naslov)

2. Realizirajte podano funkcijo  $f$  z eno 4-bitno aritmetično-logično enoto (ALU). Morebitne negacije vhodnih spremenljivk izvedite z ALU.

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

3. Prikažite sintezo 3-bitnega sinhronega števca *navzdol* po Grayevi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2$ ,  $Q_1$ ,  $Q_0$ ). Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Realizirajte avtomat končnih stanj, ki ima dva vhoda  $w_1$  in  $w_2$  in izhod  $z$ . Namen avtomata končnih stanj je primerjava zaporedja vhodnih signalov  $w_1$  in  $w_2$ : Avtomat postavi izhod  $z=1$ , če je štiri predhodne periode signala ure veljalo  $w_1=w_2$ , sicer je izhod  $z=0$ . Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhodov in izhoda.

	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
$w_1$	0	1	1	0	1	1	1	0	0	0	1	1	0
$w_2$	1	1	1	0	1	0	1	0	0	0	1	1	1
$z$	—	0	0	0	0	1	0	0	0	0	1	1	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

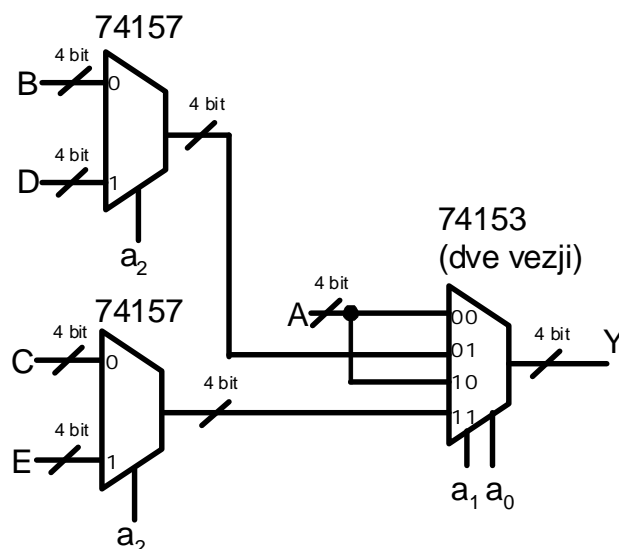
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

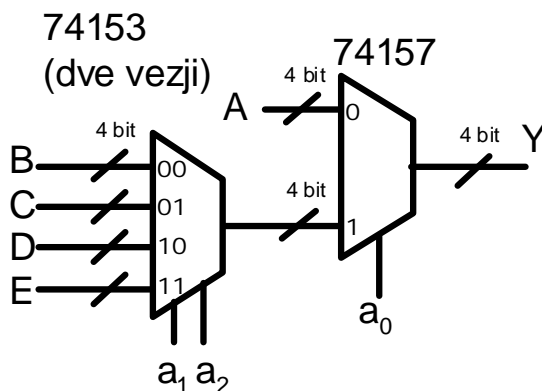
Naloga zahteva, da realiziramo izbiralnik MUX 8/1, ki je 4-biten, saj izbiramo med petimi 4-bitnimi vhodnimi vodili A, B, C, D, E. Iz tabele delovanja se vidi, da je vodilo A izbrano za vse kombinacije naslovov, pri katerih je  $a_1 a_0 = 10$  ali  $a_1 a_0 = 00$ . Te kombinacije lahko izvedemo z dvema vezjema 74153; vsako od teh vezij vsebuje dva izbiralnika 4/1 in imata skupno naslovno vodilo.

Naslovno vodilo  $a_1$   $a_0$  vežemo soležno:  $a_1$  prvega vezja in drugega 74153 skupaj in  $a_0$  prvega 74153 in drugega 74153 skupaj. Tako dobimo 4-bitni izbiralnik 4/1. Na drugem nivoju realiziramo 4-bitna izbiralnika 2/1, pri katerem vežemo na naslovni vhod  $a_2$ . Vsakega od teh izbiralnikov realiziramo z enim vezjem 74157, ki vsebuje štiri vezja 1-bitnih izbiralnikov 2/1. Vsi izbiralniki znotraj 74157 imajo že skupni naslov povezan, torej vežemo samo vzporedno obe vezji 74157, kot je narisano na sliki. Na eno vezje 74157 vežemo 4-bitni vhod B oz. D, na vezje 74157 vežemo 4-bitni vhod C oz. E. Vendar ta rešitev vsebuje 4 vezja, zato jo poskusimo dodatno minimizirati.



Ponuja se nam tudi druga možnost izvedbe, pri kateri na prvem nivoju uporabimo en 4-bitni 2/1 izbiralnik, na drugem nivoju pa dva 4-bitna 4/1 izbiralnika.

Realizacija posameznih izbiralnikov je enaka kot pri prvi rešitvi.



**Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.**

**Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSS, UNI).**

**Rezultati bodo objavljeni na:** <https://estudent.fri.uni-lj.si>

Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki MDNO.

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji so zato primerne čimbolj nenormalne oblike (večnivojske oblike), samo da vsebujejo operatorje ene vrste. Podana funkcija je v MDNO, zato za neposredno realizacijo s 4-bitno ALU ni primerna, saj vsebuje operaciji AND in OR – torej bi za realizacijo rabili najmanj dve aritmetični–logični enoti in tretjo za izvedbo inverterjev. Funkcijo MDNO prevedemo na operator enega tipa – operator NAND, kar pomeni obliko SNO (Sheffer–jeva normalna oblika funkcije):

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

Najprej pri prvih dveh členih izpostavimo člen  $d$ , saj petih operacij z eno 4 bitno ALU ne moremo izvesti.

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = \overline{\overline{(a \cdot b + \bar{c})} \cdot d + \bar{e}}$$

Za pretvorbo v SNO nad vsemi konjunkcijami izvedemo dvojno negacijo. Nad členom v oklepaju uporabimo De Morganov teorem, da dobimo izražavo z NAND operatorjem.

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{(a \cdot b)} \cdot c \right)} \cdot d + \bar{e}}$$

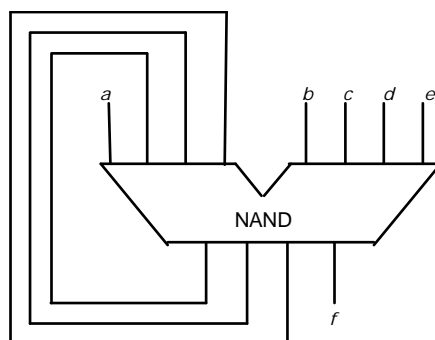
Podobno storimo še enkrat:

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{\left( \overline{(a \cdot b)} \cdot c \right)} \cdot d \cdot e \right)}}$$

Dobljene NAND operatorje predstavimo z oklepaji.

$$f(a,b,c,d,e) = \left( \left( \left( a \uparrow b \right) \uparrow c \right) \uparrow d \right) \uparrow e$$

Narišemo realizacijo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzdol po Gray–evi kodi. Desetiška števna sekvenca po 3–bitni Grayevi kodi se glasi:

... 0, 4, 5, 7, 6, 2, 3, 1, 0, ...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T–FF		
$Q_2$	$Q_1$	$Q_0$	$Q_2$	$Q_1$	$Q_0$	$T_2$	$T_1$	$T_0$
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1
0	1	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	0	1

Iz tabele prehajanja stanj števca določimo enačbe T–FF:

Za  $T_0$  narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

	$Q_2$			
$Q_1$	0	1	0	1
	1	0	1	0
	$Q_0$			

$$T_0 = Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za  $T_1$  narišemo Veitchev diagram

	$Q_2$			
$Q_1$	0	0	1	0
	0	1	0	0
	$Q_0$			

Za  $T_1$  sledi:

$$T_1 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_1 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

Operacija v oklepajih je XOR, zato enačbo lahko poenostavimo v:

$$T_1 = (Q_2 \oplus Q_1) \cdot Q_0$$

In še za  $T_2$ :

	$Q_2$			
$Q_1$	1	0	0	0
	0	0	0	1
	$Q_0$			

Za  $T_2$  sledi:

$$T_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot \overline{Q_1} + Q_2 \cdot Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR (za dve spremenljivki je to ekvivalenca), zato enačbo lahko poenostavimo v:

$$T_2 = (\overline{Q_2 \oplus Q_1}) \cdot Q_0$$



#### Rešitev 4. naloge:

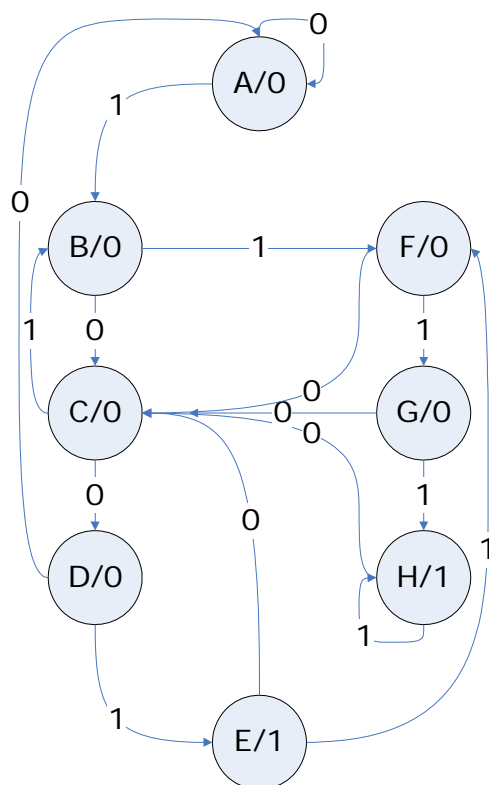


Diagram stanj začnemo risati tako, da najprej narišemo začetno stanje (A) v katerega se vračamo vedno, kadar sekvenca ne bo podobna tisti, ki jo zaznavamo (1111 oz. 1001). V tem stanju vztrajamo toliko časa, dokler je na vhodu '0', saj se nobena od sekvenc ne začneja z '0'. Ko pride na vhod prva '1', preidemo v drugo stanje (B), v katerem imamo dve možnosti, saj se sekvenci razlikujeta na drugem mestu: Če na vhod tega stanja pride '0', bomo zaznavali sekvence tipa '10XX', če pa pride '1', potem pa sekvence tipa '11XX'. Recimo, da v stanju B na vhod pride '0', torej napredujemo v stanje C. V tem stanju se ponovno lahko pojavi '0' – torej bi bila na vhodu sekvenca

tipa '100X', kar bi nas vodilo do naslednjega stanja D. Če pa se pojavi na vhodu logična '1' je sicer sekvenca napačna, vendar moramo to predstaviti tako kot da je na vhod prišla že prva '1' od sekvence - naloga namreč pravi, da se sekvence lahko med seboj prekrivajo. S podobnim načinom razmišljanja narišemo naslednje stanje E, v katero napredujemo iz D samo, ko vanj pride '1', kar pomeni da smo v tem stanju zaznali sekvenco "1001", zato je v tem stanju izhod vezja enak '1'. Stanja F, G in H služijo pomnjenju koliko enic je prišlo na vhod vezja in sicer: stanje F pomenita "11" na vhodu vezja, stanje G tri enice in zadnje stanje H četrto enico sekvence. Slednje stanje vztraja, dokler je na vhodu '1', saj tako rešimo prekrivanje vzorca '1111'. Če pa v kateremkoli od stanj F, G in H pride na vhod '0', se postavimo v stanje C, saj to stanje pomeni, da je na vhodu sekvenca '10XX'.

Tabela prehajanja stanj:

trenutno stanje		naslednje stanje		izhod
pomen stanja	koda stanja	w=0	w=1	z
začetno stanje	A	A	B	0
prva enica '1001' ali '1111' na vhodu	B	C	F	0
prva ničla '1001' na vhodu	C	D	B	0
druga ničla '1001' na vhodu	D	A	E	0
druga enica '1001' na vhodu	E	C	F	1
druga enica '1111' na vhodu	F	C	G	0
tretja enica '1111' na vhodu	G	C	H	0
četrta enica '1111' na vhodu	H	C	H	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

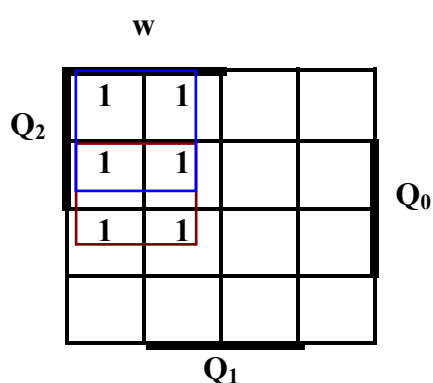
Za izvedbo bomo rabili najmanj 3 FF, saj je stanj 8. Izberemo zaporedno kodiranje stanj se pravi A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.

Vzbujalna tabela:

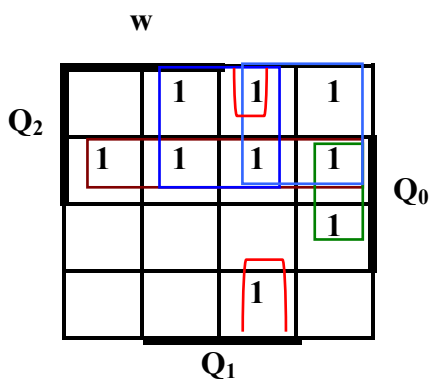
trenutno stanje				naslednje stanje						izhod
				w=0			w=1			
	Q <sub>2</sub> (t)	Q <sub>1</sub> (t)	Q <sub>0</sub> (t)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	z
A	0	0	0	0	0	0	0	0	1	0
B	0	0	1	0	1	0	1	0	1	0
C	0	1	0	0	1	1	1	0	1	0
D	0	1	1	0	0	0	1	0	0	0
E	1	0	0	0	1	0	1	0	1	1
F	1	0	1	0	1	0	1	1	0	0
G	1	1	0	0	1	0	1	1	1	0
H	1	1	1	0	1	0	1	1	1	1

Iz vzbujalne tabele sestavimo tri Veitcheve diagrame:

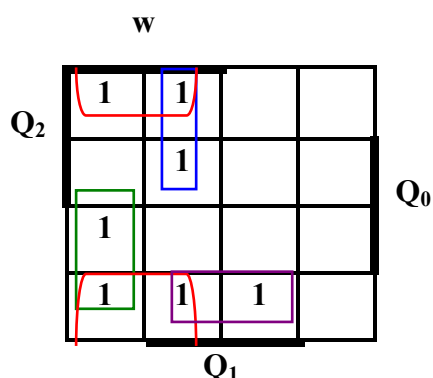
$Q_2(t+1)$ :



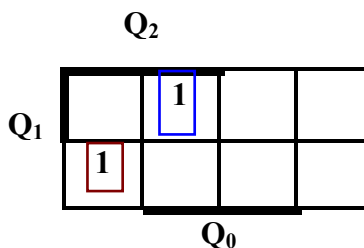
$Q_1(t+1)$ :



$Q_0(t+1)$ :



z:



Funkcije v Veitchevih diagramih minimiziramo in dobimo enačbe za realizacijo s pomočjo D flip-flopov. Realizacija s pomočjo D flip-flopov je najbolj enostavna, saj je enačba D flip-flopa:

$$D = Q(t+1) \quad (3.1)$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

kar pomeni, da lahko iz minimizacije Veitchevih diagramov naslednjih stanj zapišemo enačbe za vhode D flip-flopov:

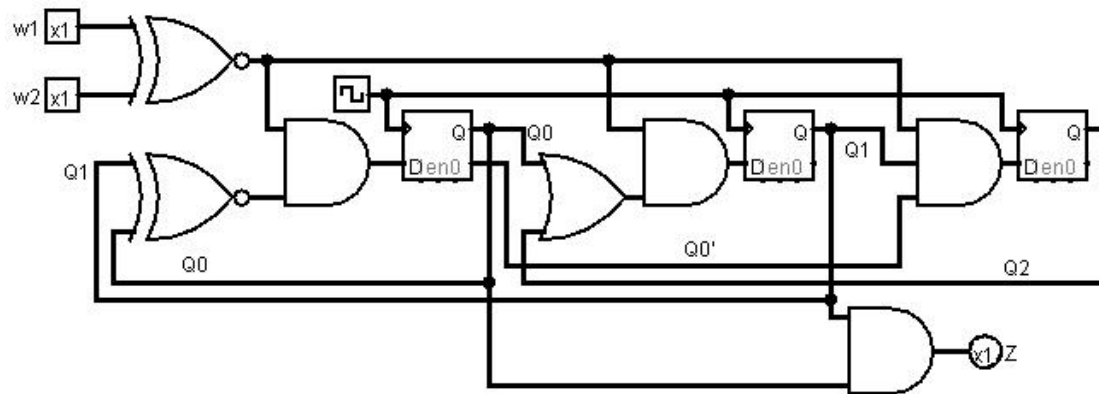
$$\begin{aligned}
 D_2 &= Q_2(t+1) = w \cdot Q_2(t) + \bar{w} \cdot Q_0(t) = w \cdot (Q_2(t) + Q_0(t)) \\
 D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + \bar{Q}_1(t) \cdot Q_0(t) + Q_1(t) \cdot \bar{Q}_0(t)) = \\
 D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + Q_1(t) \oplus Q_0(t)) \\
 D_0 &= Q_0(t+1) = w \cdot \bar{Q}_0(t) + w \cdot \bar{Q}_2(t) \cdot \bar{Q}_1(t) + w \cdot Q_2(t) \cdot Q_1(t) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t) = \\
 D_0 &= Q_0(t+1) = w \cdot (\bar{Q}_0(t) + \bar{Q}_2(t) \oplus Q_1(t)) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t)
 \end{aligned} \tag{3.2}$$

Izhod z lahko zapišemo kot:

$$z = Q_2(t) \cdot (Q_1(t) \cdot Q_0(t) + \bar{Q}_1(t) \cdot \bar{Q}_0(t)) = Q_2(t) \cdot \overline{Q_1(t) \oplus Q_0(t)} \tag{3.3}$$

Vezje avtomata narišemo iz enačb (3.2) in (3.3).

Opis delovanja in vezje avtomata, ki primerja enakost znotraj 4 period signala ure je v predlogah vaj na domači strani predmeta v imeniku Logisim\fsm\fsm\_four\_periodes\_of\_equality.circ



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

12. 12. 2013

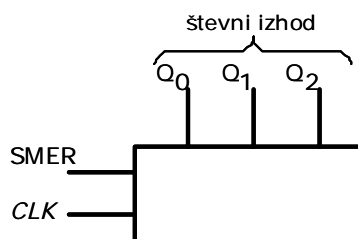
1. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5-7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

2. Realizirajte podano funkcijo  $f$  z eno 4-bitno aritmetično-logično enoto (ALU). Morebitne negacije vhodnih spremenljivk izvedite z ALU.

$$f(a, b, c, d, e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

3. Prikažite sintezo sinhronega dvosmernega 3-bitnega števca z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov. Števec ima vhod SMER, ki določa smer štetja: Če je SMER='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

# Rešitev 1. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

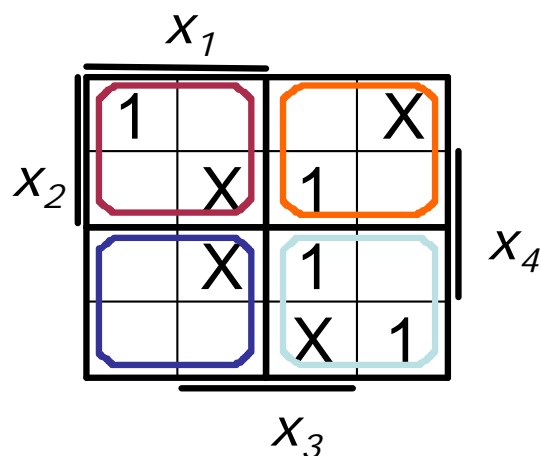
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f='0'$  za vse maksterme in  $f='X'$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f='1'$  in preberemo pri katerih mintermih je  $f='1'$  oz.  $f='X'$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

Dobljeno funkcijo vrišemo v Veitch-ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev-em diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:



V zgornjem Veitch-evem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2="11"$ , oranžni kvadrat ko bo  $x_1x_2="01"$ , temno modri ko bo  $x_1x_2="10"$  in svetlo modri ko bo  $x_1x_2="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3'x_4'$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4$ . Funkcija bo torej  $x_3'x_4' + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

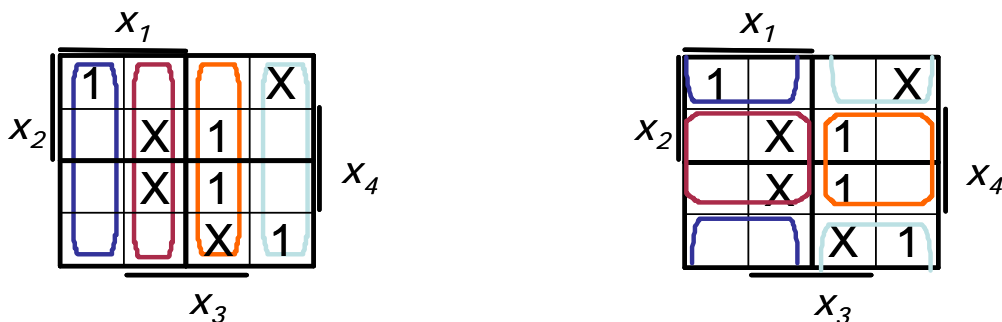
Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram,

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

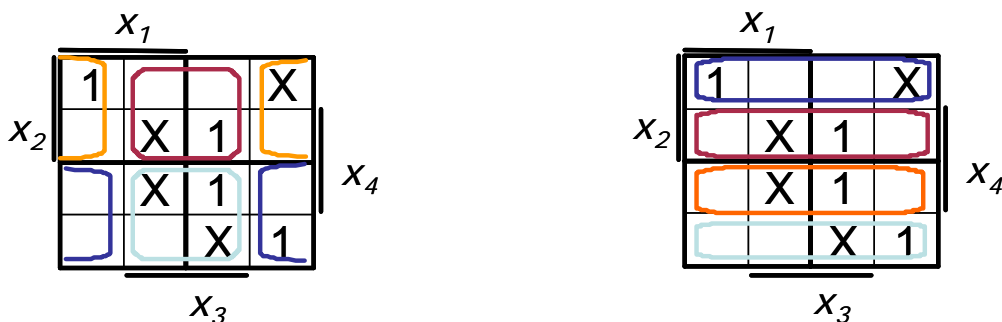
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

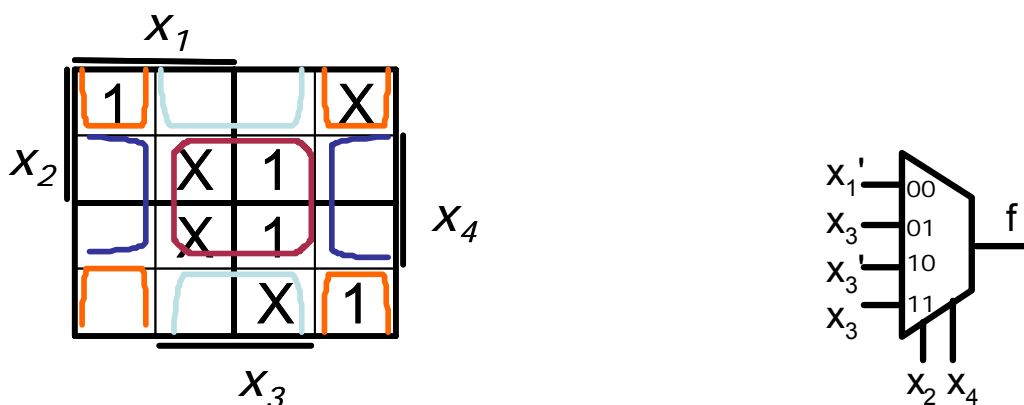
če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo drugače kvadratov, ki vsebujejo konstante (samo '1' ali samo '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .



Nato izberemo naslovni spremenljivki  $x_2$  in  $x_3$ , (levi diagram) in  $x_2$  in  $x_4$  (desni diagram). Pri razvoju po  $x_2$  in  $x_3$  imamo pri  $x_2x_3="00"$  najneugodnejšo funkcijo (moder), saj vsebuje eno samo '1'. Pri razvoju po  $x_2$  in  $x_4$  nikjer nimamo osamljene '1', zato se dajo funkcijski ostanki enostavno realizirati, če vse redundance postavimo na '1'.



Zadnja kombinacija naslovnih vhodov je  $x_3$  in  $x_4$ . Pri razvoju po  $x_3$  in  $x_4$  imamo pri  $x_3x_4="00"$  najneugodnejšo funkcijo (oranžen), saj dve '1' opišemo s funkcijo ekvivalence. Možno rešitev torej predstavlja kombinacija naslovnih vhodov  $x_2$  in  $x_4$ .



Vezje izbiralnika je v predlogah avditornih vaj na domači strani predmeta:  
Logisim\MUX\mux\_4\_1\_f\_V\_0\_3\_7\_12\_in\_Vx\_2\_4\_11\_15.circ

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki MDNO.

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji so zato primerne čimbolj nenormalne oblike (večnivojske oblike), samo da vsebujejo operatorje ene vrste. Podana funkcija je v MDNO, zato za neposredno realizacijo s 4-bitno ALU ni primerna, saj vsebuje operaciji AND in OR – torej bi za realizacijo rabili najmanj dve aritmetični–logični enoti in tretjo za izvedbo inverterjev. Funkcijo MDNO prevedemo na operator enega tipa – operator NAND, kar pomeni obliko SNO (Sheffer–jeva normalna oblika funkcije):

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

Najprej pri prvih dveh členih izpostavimo člen  $d$ , saj petih operacij z eno 4 bitno ALU ne moremo izvesti.

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = \overline{\overline{(a \cdot b + \bar{c}) \cdot d + \bar{e}}}$$

Za pretvorbo v SNO nad vsemi konjunkcijami izvedemo dvojno negacijo. Nad členom v oklepaju uporabimo De Morganov teorem, da dobimo izražavo z NAND operatorjem.

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{(a \cdot b) \cdot c} \right) \cdot d + \bar{e}}}$$

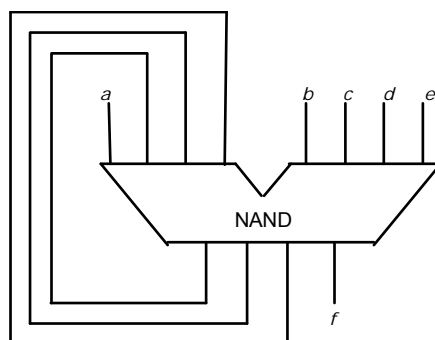
Podobno storimo še enkrat:

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{\overline{(a \cdot b) \cdot c}} \right) \cdot d \cdot e}}$$

Dobljene NAND operatorje predstavimo z oklepaji.

$$f(a,b,c,d,e) = \left( \left( \left( a \uparrow b \right) \uparrow c \right) \uparrow d \right) \uparrow e$$

Narišemo realizacijo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF: Iz stolpca T<sub>0</sub> se vidi, da je T<sub>0</sub>='1'. Iz stolpca T<sub>1</sub> se vidi, da se ponavlja

vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T <sub>1</sub>
0	Q <sub>0</sub>
1	Q <sub>0</sub> '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

Za T<sub>2</sub> se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

	SMER				
	1	0	0	0	
Q <sub>2</sub>	0	0	1	0	
	0	0	1	0	Q <sub>0</sub>
	1	0	0	0	
	Q <sub>1</sub>				
	$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$				

V enačbi za T<sub>2</sub> poiščemo podobnosti z enačbo za T<sub>1</sub>: Enačba za T<sub>1</sub> vsebuje konjunkciji SMER·Q<sub>0</sub>' in SMER'·Q<sub>0</sub>, ki sta vsebovani tudi v enačbi za T<sub>2</sub>, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191<sup>1</sup>. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74191>



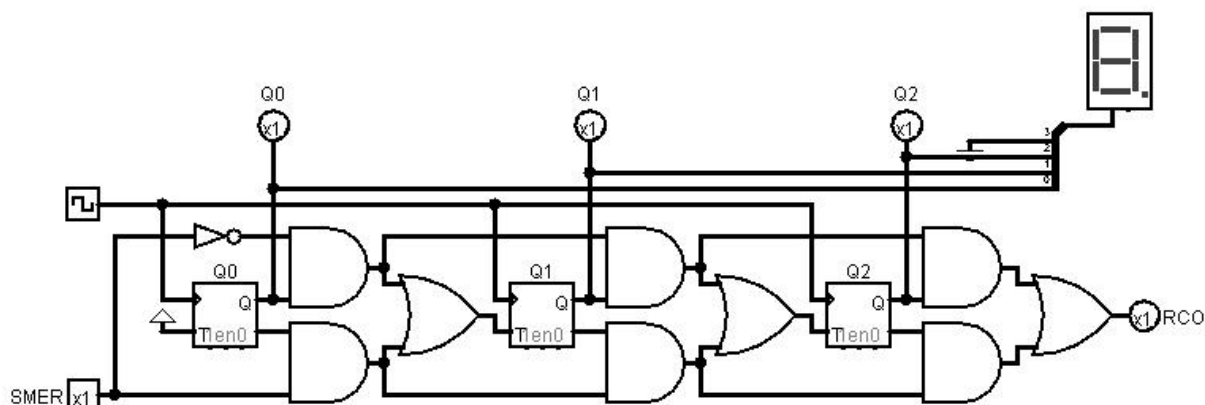
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca (RCO – oz. ripple carry out, včasih tudi TC – terminal count). RCO je signal, ki postane '1' ob prehodu iz najvišjega stanja števca (v našem primeru je to "111") v stanje "000" pri štetju navzgor in ob prehodu "000" v najvišje stanje števca pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števec vežemo kaskadno – torej da signal RCO vežemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu  $Q_2$ , kot kaže spodnja slika:



Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_up\_down\_3\_bit\_using\_T\_FF.circ

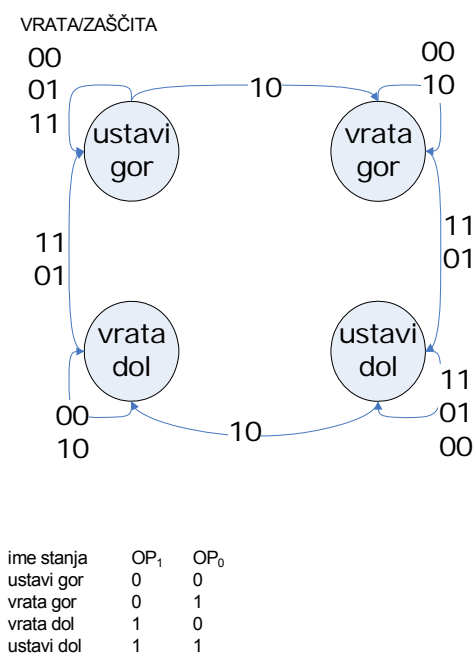
Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

- desetiški (BCD) števci, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161<sup>2</sup>.

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74161>

#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanji "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji

"11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

24. 01. 2014

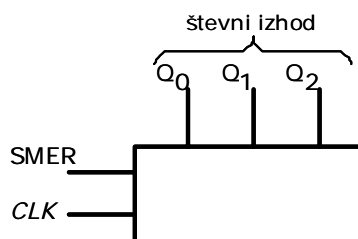
1. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5-7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

2. Realizirajte podano funkcijo  $f$  z eno 4-bitno aritmetično-logično enoto (ALU). Morebitne negacije vhodnih spremenljivk izvedite z ALU.

$$f(a, b, c, d, e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

3. Prikažite sintezo sinhronega dvosmernega 3-bitnega števca z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov. Števec ima vhod SMER, ki določa smer štetja: Če je SMER='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

# Rešitev 1. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

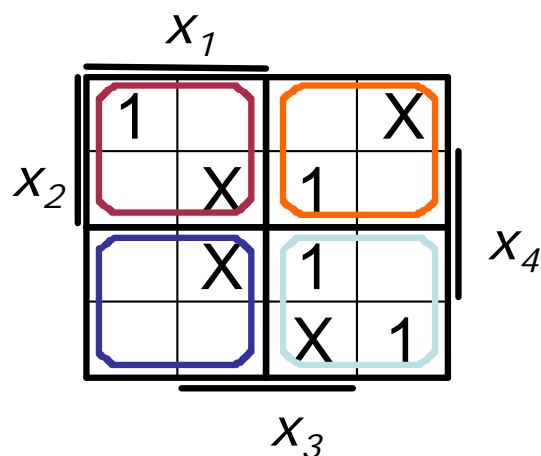
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

Dobljeno funkcijo vrišemo v Veitch-ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev-em diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:



V zgornjem Veitch-evem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2=11$ , oranžni kvadrat ko bo  $x_1x_2=01$ , temno modri ko bo  $x_1x_2=10$  in svetlo modri ko bo  $x_1x_2=00$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3x_4$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4 + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

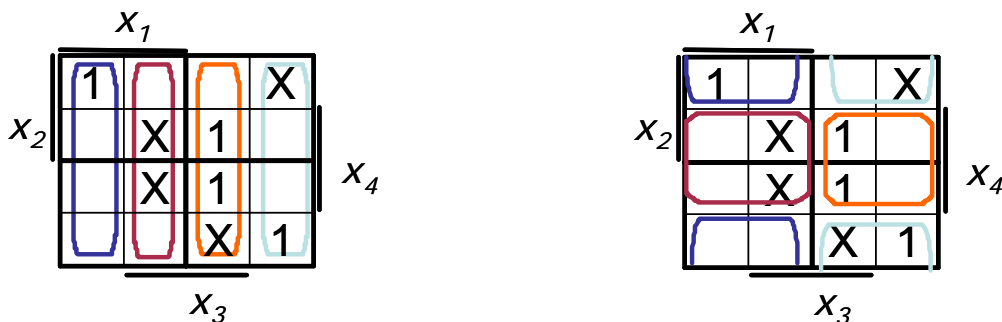
Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram,

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

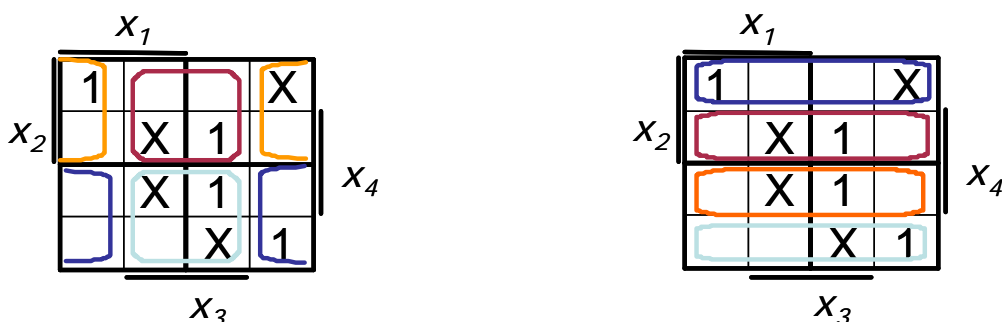
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

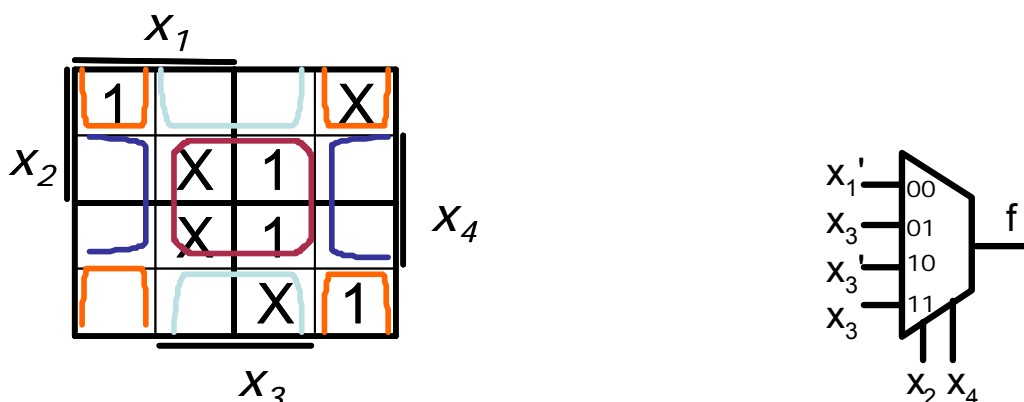
če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo drugače kvadratov, ki vsebujejo konstante (samo '1' ali samo '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .



Nato izberemo naslovni spremenljivki  $x_2$  in  $x_3$ , (levi diagram) in  $x_2$  in  $x_4$  (desni diagram). Pri razvoju po  $x_2$  in  $x_3$  imamo pri  $x_2x_3="00"$  najneugodnejšo funkcijo (moder), saj vsebuje eno samo '1'. Pri razvoju po  $x_2$  in  $x_4$  nikjer nimamo osamljene '1', zato se dajo funkcijski ostanki enostavno realizirati, če vse redundance postavimo na '1'.



Zadnja kombinacija naslovnih vhodov je  $x_3$  in  $x_4$ . Pri razvoju po  $x_3$  in  $x_4$  imamo pri  $x_3x_4="00"$  najneugodnejšo funkcijo (oranžen), saj dve '1' opišemo s funkcijo ekvivalence. Možno rešitev torej predstavlja kombinacija naslovnih vhodov  $x_2$  in  $x_4$ .



Vezje izbiralnika je v predlogah avditornih vaj na domači strani predmeta:  
Logisim\MUX\mux\_4\_1\_f\_V\_0\_3\_7\_12\_in\_Vx\_2\_4\_11\_15.circ

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki MDNO.

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji so zato primerne čimbolj nenormalne oblike (večnivojske oblike), samo da vsebujejo operatorje ene vrste. Podana funkcija je v MDNO, zato za neposredno realizacijo s 4-bitno ALU ni primerna, saj vsebuje operaciji AND in OR – torej bi za realizacijo rabili najmanj dve aritmetični–logični enoti in tretjo za izvedbo inverterjev. Funkcijo MDNO prevedemo na operator enega tipa – operator NAND, kar pomeni obliko SNO (Sheffer–jeva normalna oblika funkcije):

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

Najprej pri prvih dveh členih izpostavimo člen  $d$ , saj petih operacij z eno 4 bitno ALU ne moremo izvesti.

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = \overline{\overline{(a \cdot b + \bar{c}) \cdot d + \bar{e}}}$$

Za pretvorbo v SNO nad vsemi konjunkcijami izvedemo dvojno negacijo. Nad členom v oklepaju uporabimo De Morganov teorem, da dobimo izražavo z NAND operatorjem.

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{(a \cdot b) \cdot c} \right) \cdot d + \bar{e}}}$$

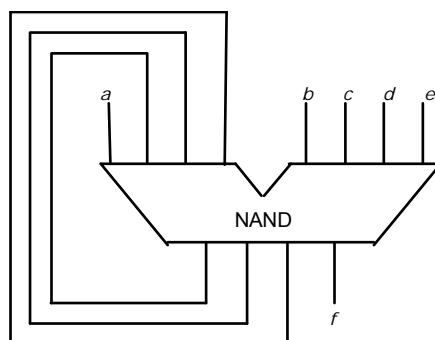
Podobno storimo še enkrat:

$$f(a,b,c,d,e) = \overline{\overline{\left( \overline{\overline{(a \cdot b) \cdot c}} \right) \cdot d \cdot e}}$$

Dobljene NAND operatorje predstavimo z oklepaji.

$$f(a,b,c,d,e) = \left( \left( \left( a \uparrow b \right) \uparrow c \right) \uparrow d \right) \uparrow e$$

Narišemo realizacijo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števcov, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF: Iz stolpca T<sub>0</sub> se vidi, da je T<sub>0</sub>='1'. Iz stolpca T<sub>1</sub> se vidi, da se ponavlja

vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T <sub>1</sub>
0	Q <sub>0</sub>
1	Q <sub>0</sub> '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

Za T<sub>2</sub> se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

	SMER				
	1	0	0	0	
Q <sub>2</sub>	0	0	1	0	
	0	0	1	0	Q <sub>0</sub>
	1	0	0	0	
	Q <sub>1</sub>				
	$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$				

V enačbi za T<sub>2</sub> poiščemo podobnosti z enačbo za T<sub>1</sub>: Enačba za T<sub>1</sub> vsebuje konjunkciji SMER·Q<sub>0</sub>' in SMER'·Q<sub>0</sub>, ki sta vsebovani tudi v enačbi za T<sub>2</sub>, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191<sup>1</sup>. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74191>

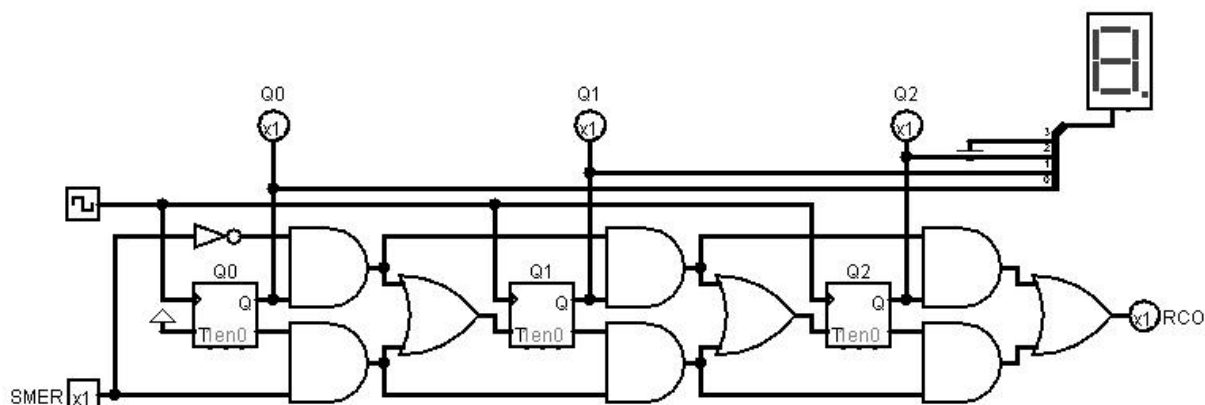
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca (RCO – oz. ripple carry out, včasih tudi TC – terminal count). RCO je signal, ki postane '1' ob prehodu iz najvišjega stanja števca (v našem primeru je to "111") v stanje "000" pri štetju navzgor in ob prehodu "000" v najvišje stanje števca pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števec vežemo kaskadno – torej da signal RCO vežemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu  $Q_2$ , kot kaže spodnja slika:



Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_up\_down\_3\_bit\_using\_T\_FF.circ

Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

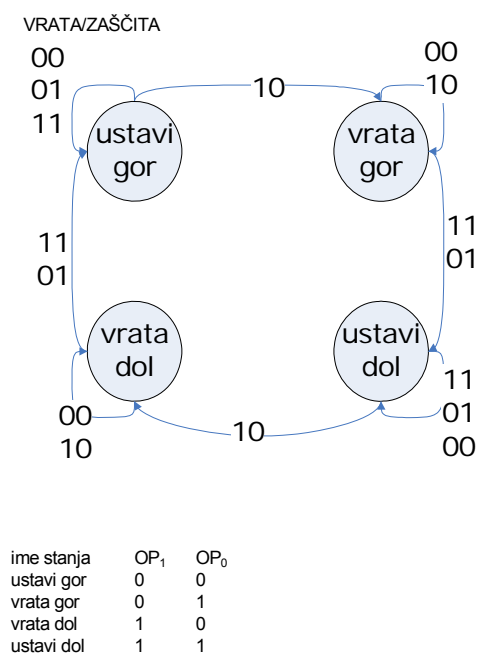
- desetiški (BCD) števci, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161<sup>2</sup>.

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74161>



#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanji "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji

"11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

# RAZVOJ DIGITALNIH SISTEMOV

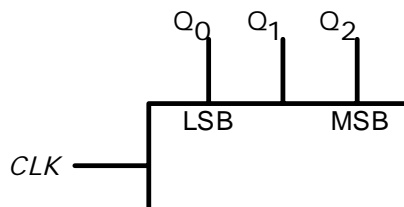
Izpit

07. 02. 2014

1. Zgradite vezje, katerega izhod postane '1', ko se na vhodu pojavi eno izmed praštevil (2, 3, 5, 7, 11, 13). Števila na vhodu so kodirana s 4-bitno Gray-evo kodo. Za realizacijo uporabite samo NAND vrata. Kolikšna je COST funkcija realiziranega vezja?
2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:
  - $f_1 = x_1 \oplus x_2$
  - $f_2 = x_1 \cdot x_2 \cdot x_3$
  - $f_3 =$  funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh.

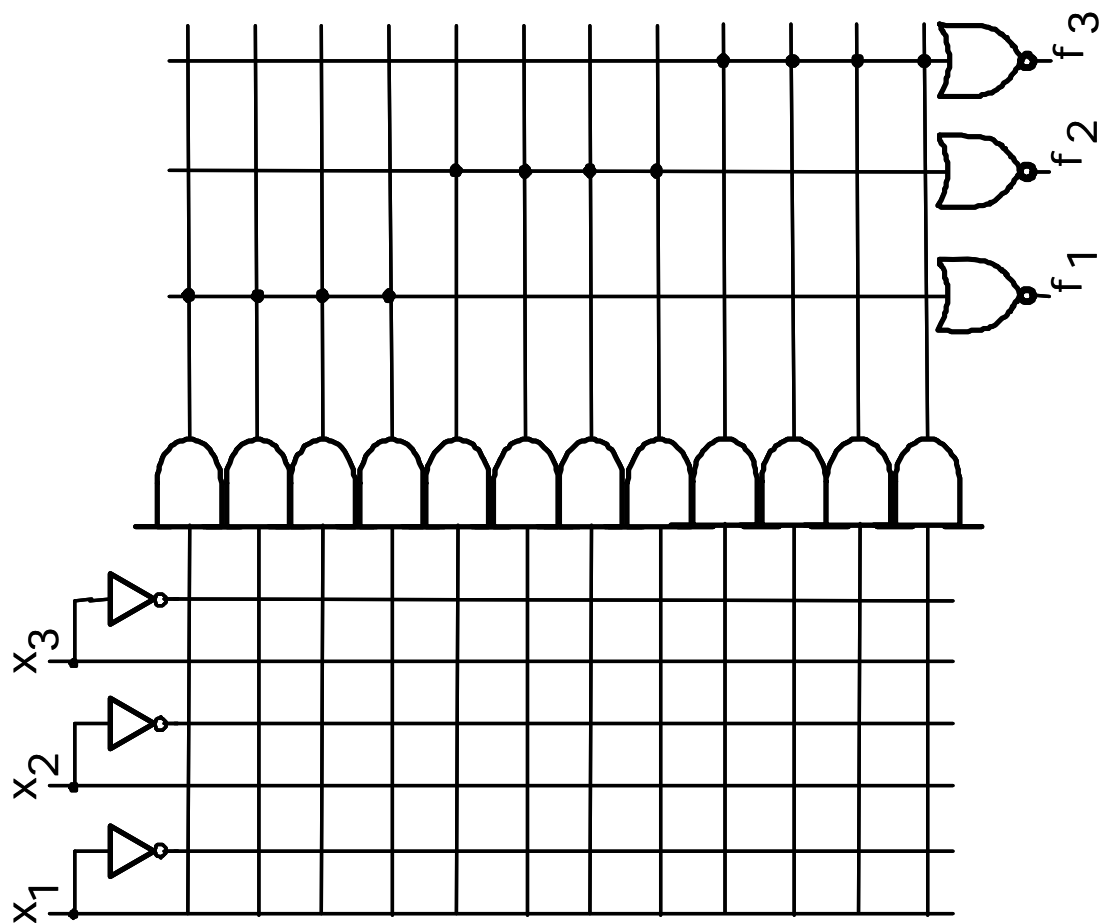
Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko. Vezje PAL je narisano na hrbtni strani izpita.

3. Prikažite sintezo sinhronnega 3-bitnega števca navzdol z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov ter vezje narišite. Imena signalov so razvidna iz spodnje slike.

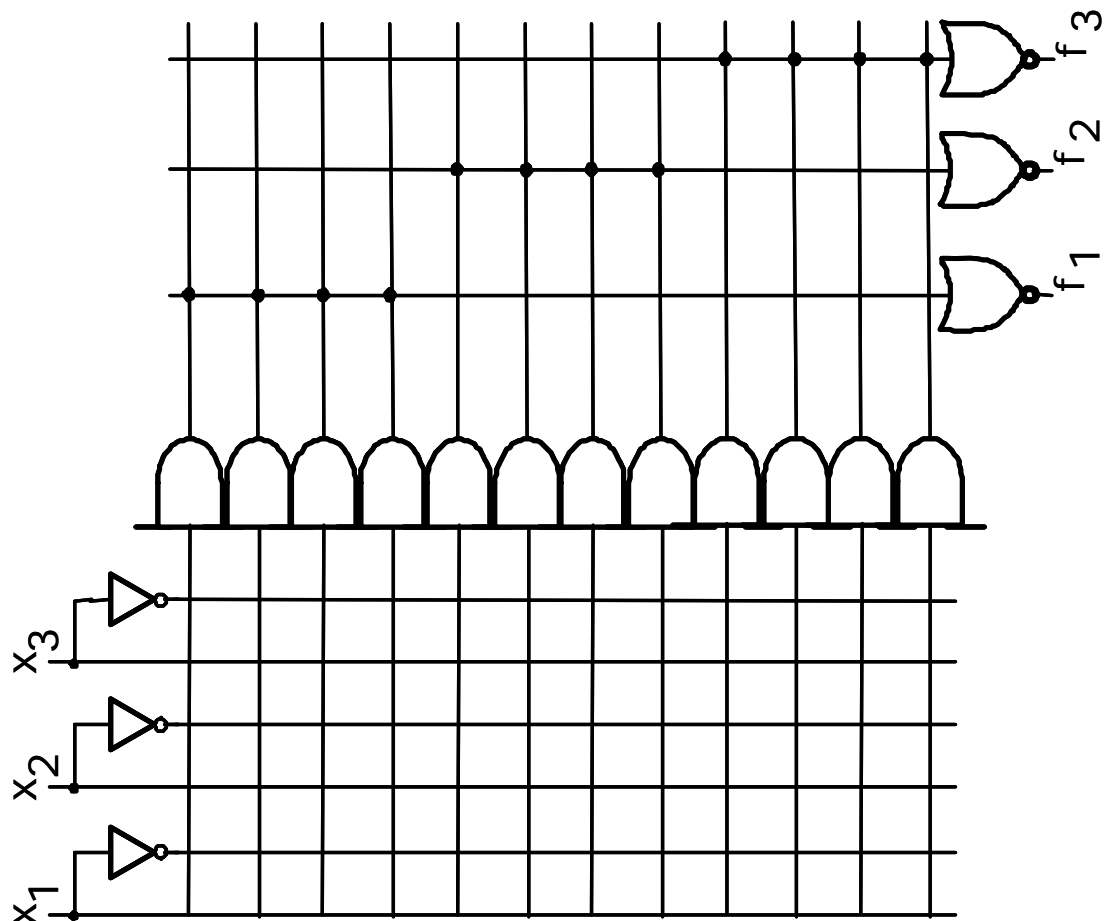


4. Narišite diagram stanj za avtomat končnih stanj, ki ima vhod w in izhod z. Avtomat končnih stanj postavi izhod  $z=1$ , ko se na vhodu pojavi zaporedje "110" ali "101", sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda. Tip avtomata je razviden iz podanega časovnega zaporedja.

CLK	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
w	0	1	1	0	1	1	0	1	1	1	0	0	0
z	—	0	0	1	1	0	1	1	0	0	1	0	0



Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

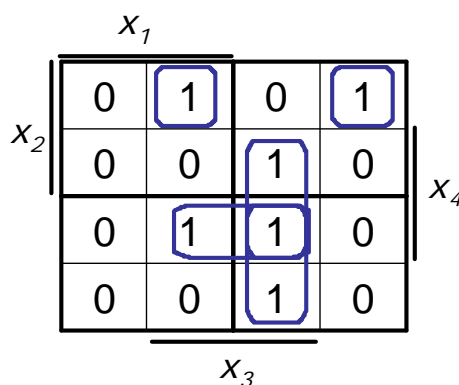
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

S 4-bitno Grayevo kodo lahko kodiramo 16 števil (0 - 15). Najprej zapišemo pravilnostno tabelo v kateri desetiška števila kodiramo v Grayevi kodi (npr.  $7_{10}=0100_{\text{Gray}}$ ). Izhod mora biti '1' vsakič, ko se na vhodu pojavi praštevilo. Izpisano funkcijo nato izpišemo v Veitch-ev diagram, od koder bomo lahko izvajali postopek minimizacije v PDNO, iz te oblike pa bomo prešli na PSNO (NAND operatorji).

št.	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	1	1
3	0	0	1	0	1
4	0	1	1	0	0
5	0	1	1	1	1
6	0	1	0	1	0
7	0	1	0	0	1
8	1	1	0	0	0
9	1	1	0	1	0
10	1	1	1	1	0
11	1	1	1	0	1
12	1	0	1	0	0
13	1	0	1	1	1
14	1	0	0	1	0
15	1	0	0	0	0



$$f_{MDNO} = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}$$

Da bi iz PDNO prešli na realizacijo s samimi NAND vrati, funkcijo v PDNO dvakrat negiramo in uporabimo de Morgan-ov teorem:

$$\overline{\overline{f_{MDNO}}} = \overline{\overline{x_1 \cdot \overline{x_2} \cdot x_3 + \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}}}$$

$$f_{PSNO} = \overline{\overline{(x_1 \cdot \overline{x_2} \cdot x_3)} \cdot \overline{(x_2 \cdot x_3 \cdot x_4)} \cdot \overline{(x_1 \cdot x_3 \cdot x_4)} \cdot \overline{(x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4})} \cdot \overline{(x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4})}}$$

Za nastalo PSNO moramo določiti COST vezja (brez inverterjev):

OBLIKA	VRAT	VHODOV	COST
PSNO	6	22	28

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$f_2 = \overline{x_1 \cdot x_2 \cdot x_3}$$

$$\overline{f_2} = \overline{x_1 + x_2 + x_3}$$

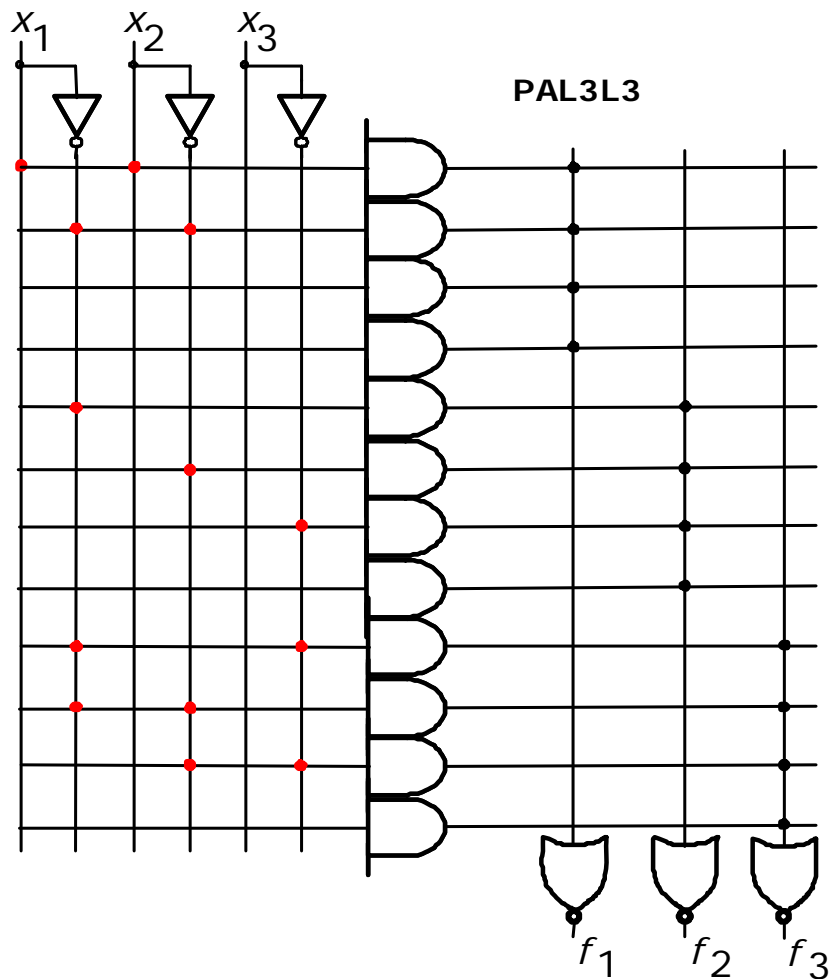
Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

$$\overline{f_3} = \overline{x_1 \cdot x_3 + x_1 \cdot x_2 + x_2 \cdot x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vezemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6–vhodna. Vezje PAL3L3 je AND–NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch-eve diagrame za tri spremenljivke za vsak vhod T-FF, vendar ker so T-FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Iz stolpca T<sub>0</sub> se vidi:

$$T_0 = 1$$

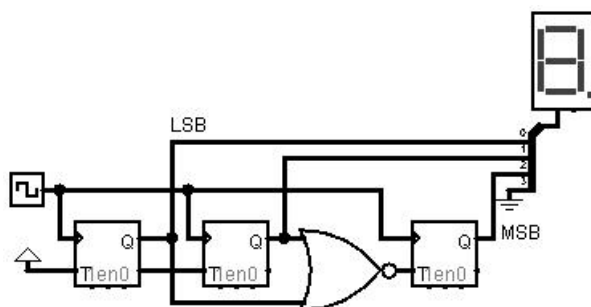
Z opazovanjem stolpcev trenutnega stanja določimo T<sub>1</sub>:

$$T_1 = \overline{Q_0}$$

Podobno lahko določimo T<sub>2</sub>:

$$T_2 = \overline{Q_0} \cdot \overline{Q_1} = \overline{Q_0 + Q_1}$$

Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_7\_0\_using\_T\_FF.circ:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

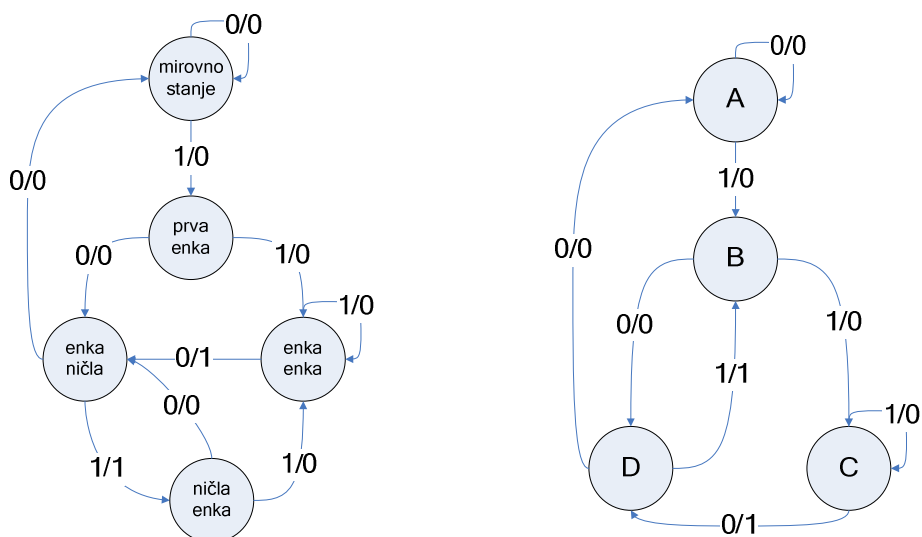
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Avtomat je Mealy–eve izvedbe, ker izhod postane '1' takoj, ko se pojavi zadnja vrednost zaznavanega zaporedja.

Zapišemo začetno stanje A, v katerem ostajamo toliko časa, dokler se ne začne ena od sekvenc, ki ju zaznavamo. Obe sekvenci se začneta z '1', zato v stanje B preidemo, ko je na vhodu prva '1'. V stanju B ne moremo ostati, saj se na vhodu lahko pojavi '0' ali '1' – v obeh primerih gre za del zaznavanega zaporedja "10X" ali "11X". Iz stanja B preidemo v stanje C, če se vmes pojavi '1', tako da v tem stanju pomeni detekcijo sekvence "11X", v stanje D pa preidemo če se pojavi na vhodu '0', kar pomeni detekcijo sekvence "10X".

Prekrivanje zaporedij: Če se v stanju C pojavi '1' na vhodu, potem gre za sekvenco "111" na vhodu – kar še vedno pomeni, da ostajamo v stanju C, saj je prekrivanje vzorcev dovoljeno. Drugače se diagram obnaša, ko smo v stanju D in pride na vhod še ena '0' – takrat smo imeli na vhodu sekvenco "100", tako da se moramo vrniti v stanje A, saj se nobena od zaznavanih sekvenc ne začneja z '0'.



Stanju "ničla enka" se lahko izognemo, saj je to stanje ekvivalentno stanju "prva enka" in dobimo izvedbo avtomata, prikazano na desni strani.



# RAZVOJ DIGITALNIH SISTEMOV

Izpit

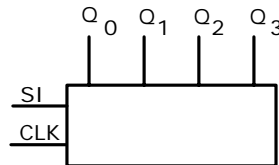
23. 06. 2014

1. Realizirajte funkcijo  $f$  s čim manj izbiralniki 4/1.

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Pretvorite število  $59_{16}$  v BCD zapis z uporabo "double dabble" algoritma.

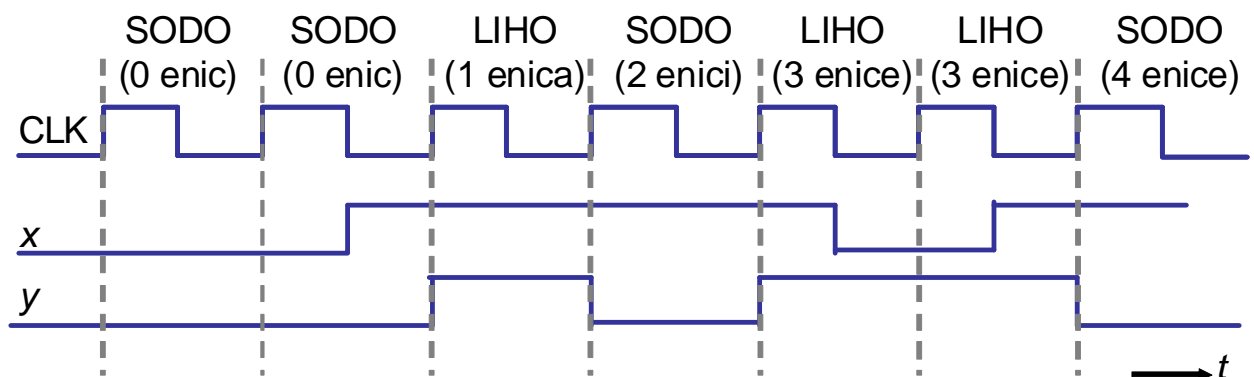
3. Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod  $SI$  (ang. serial input), in vzporedni izhod ( $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ ). Uporabite poimenovanje signalov na spodnji sliki.



4. Realizirajte generator lihe parnosti (paritete) kot avtomat končnih stanj, ki šteje število enic v serijskem zaporedju bitov  $x$  na vhodu. Izhod vezja  $y$  naj bo '1', ko je na vhodu liho število enic in '0' ko je na vhodu sodo število enic. Ob resetu avtomata je število enic na vhodu sodo (nič enic). Za realizacijo uporabite D flip-flope, prožene na sprednji rob signala ure  $CLK$ .



Primer delovanja generatorja parnosti povzema spodnja slika:



Rešitev 1. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

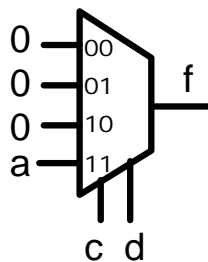
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

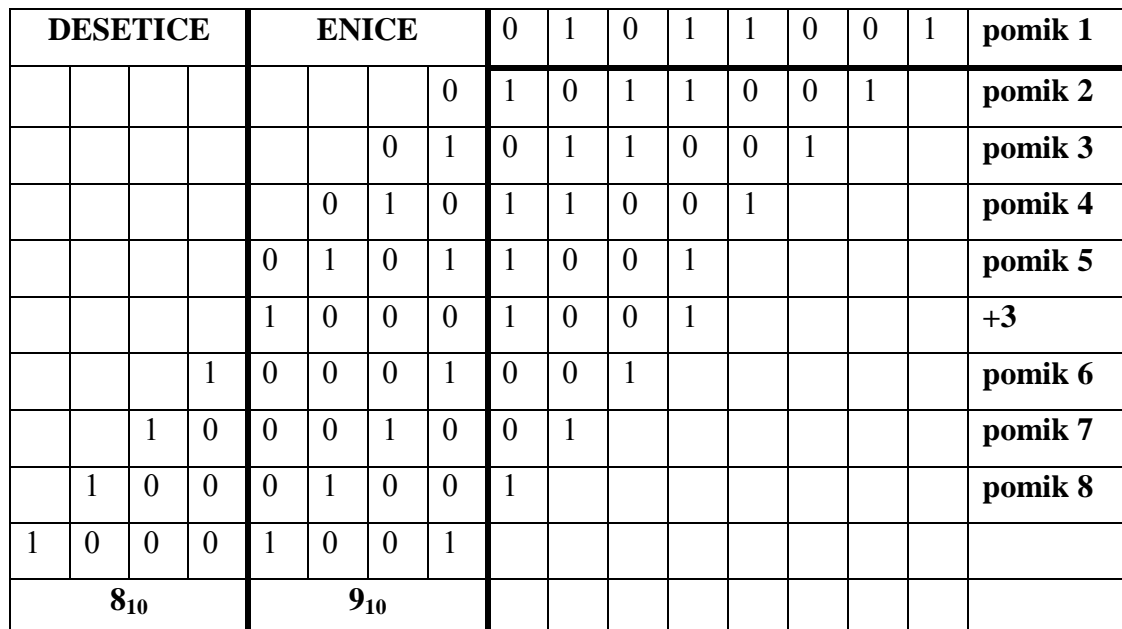
$$f_{PDNO}(a,b,c,d) = V(11,15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).



Pretvorimo šestnajstiko vrednost  $59_{16} = 89_{10}$ . Zapis posameznih števk: 0101 1001<sub>BCD</sub>. Pretvorbo po "double dabble" algoritmu opravimo po spodnjem algoritmu:



Postopek sinteze zahteva, da realiziramo avtomat končnih stanj Moore-ove izvedbe: Najprej bomo razvili diagram prehajanja stanj, ki opisuje delovanje vezja za liho preverjanje parnosti. Vezje je lahko v enem od dveh stanj: v sekvenci je bilo do tega

trenutka liho ali sodo število enic. Kadar je na vhodu 1, je potrebno preklopiti v drugo stanje. Na primer, če je bilo do tega trenutka prisotnih liho število enic in je trenutni vhod 1, potem bomo imeli sedaj sodo število enic. Če pa bo na vhodu 0, ostane v istem stanju. Narisani diagram prehajanja stanj ima dve stanji, ki označujeta trenutno število enic na vhodu – torej LIHO in SODO. Izhod zapišemo pod stanjem (LIHO='1', SODO='0'). Vrednosti na vhodu x povzročajo spreminjanje stanj, ki so označene z usmerjenimi povezavami. Če je se na vhodu pojavi '0' (ne glede na to v katerem stanju smo) ostanemo v tem stanju: Jasno – saj štejemo samo '1'. Če smo v stanju LIHO in se na vhodu pojavi '1', preidemo v SODO. Če smo v stanju SODO in se na vhodu pojavi '1', preidemo v LIHO. Povedano povzema spodnji diagram prehajanja stanj

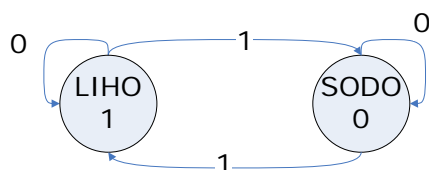


Diagram prehajanja stanj opišemo s tabelo prehajanja stanj:

vhod x	trenutno stanje Q(t)	naslednje stanje Q(t+1)
0	SODO	SODO
0	LIHO	LIHO
1	SODO	LIHO
1	LIHO	SODO

Če stanja kodiramo glede na njihov izhod LIHO '1' in SODO '0', potem dobimo novo aplikacijsko tabelo prehajanja stanj.

x	Q(t)	Q(t+1)	D	y
0	0	0	0	0
0	1	1	1	0
1	0	1	1	1
1	1	0	0	1

Iz tabele prehajanja stanj avtomata določimo enačbo za D-FF. Potrebno število FF je 1, saj sta stanji samo dve.

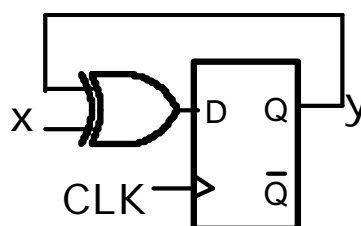
Iz aplikacijske tabele sledi:

$$D = Q(t+1)$$

$$Q(t+1) = x \cdot \overline{Q(t)} + \overline{x} \cdot Q(t) = x \oplus Q(t)$$

$$y = Q(t)$$

Izvedba vezja je:



Realizirali smo T-FF, prožen na sprednji rob signala ure.

Delovanje vezja si lahko ogledate v predlogah Logisim na domači strani predmeta:

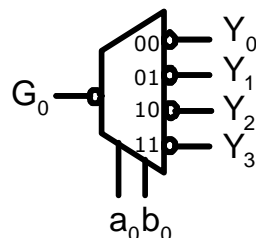
Logisim\ff\T\_ff\_using\_D\_ff\_and\_xor.circ

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 17. 09. 2014

- Realizirajte funkcijo  $f^4 = V(1, 2, 4, 7)$  z redundantnimi mintermi pri  $V_x(0, 3, 6)$  z enim TTL dekodiranjem 74139. Dekoder 74139 ima vhod za omogočenje elementa ( $G$ ) in izhode  $Y_0, Y_1, Y_2, Y_3$  v negativni logiki. Njegovo delovanje povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

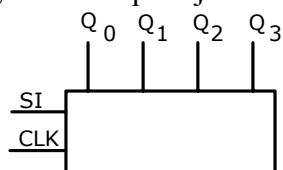


- Uporabite ROM vezje za realizacijo funkcij:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2$$

ROM vezje ima 3 naslovne spremenljivke in 2 bitno vsebino. Narišite shemo ROM vezja in v shemi označite programirane povezave oz. 'varovalke' s piko (●).

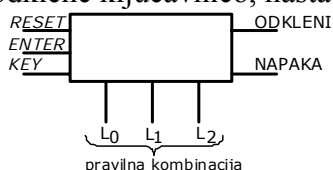
- Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod  $SI$  (ang. serial input), in vzporedni izhod ( $Q_0, Q_1, Q_2, Q_3$ ). Uporabite poimenovanje signalov na spodnji sliki.



- Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev ( $RESET$ ), ki postavlja ključavnico v začetno stanje, tipko ( $ENTER$ ) za vnos nastavljene kombinacije in preklopnik ( $KEY$ ) za vnos enega bita kombinacije. Izhod ključavnice je ( $ODKLENI$ ), ki postane '1' ko uporabnik vnese pravilno kombinacijo in izhod ( $NAPAKA$ ), ki postane '1' če je vnesena kombinacija napačna. Uporabo ključavnice povzema spodnje zaporedje:

- 1.) pritisnemo  $RESET$
- 2.) s preklopnikom  $KEY$  nastavimo bit kombinacije odklepanja
- 3.) pritisnemo  $ENTER$
- 4.) izvede se primerjava i-tega bita ( $KEY=L_i$ )
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi  $ODKLENI='1'$ , sicer se postavi  $NAPAKA='1'$ .
- 7.) ponoven pritisk na  $RESET$  nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti  $L_0, L_1$  in  $L_2$ .



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

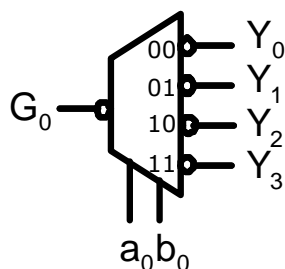
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

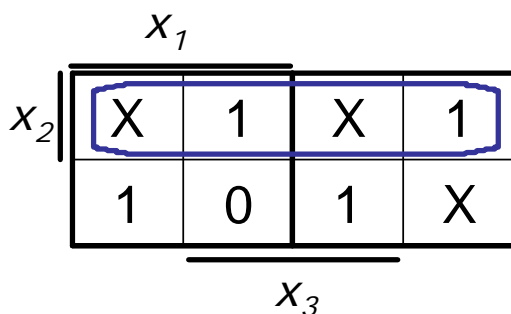
Rešitev 1. naloge:

Delovanje dekoderja 74139<sup>1</sup> povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

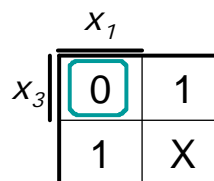


Funkcijo  $f$  narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

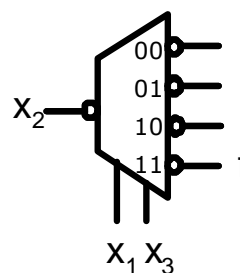


Čim imamo na voljo dekodirnik z ENABLE vhodom v negativni logiki preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '1' vključno z redundancami X.

V zgornjem Veitch–evem diagramu je to spremenljivka  $x_2$ : Namreč, če je  $x_2=1$ , potem lahko vse redundance izberemo tako, da bo  $f=1$  za vse vrednosti  $x_2=1$ . Ko določimo spremenljivko za omogočenje elementa ( $G$ ), opazujemo samo preostali del Veitch–evega diagrama. Spodnje 4 vrednosti diagrama narišemo v novem Veitch–evem diagramu 2 spremenljivk.



Dekoder ima aktivno nizke izhode, zato iz nastalega diagrama realiziramo *negacijo* funkcije  $f$  zato v Veitch–evem diagramu združujemo ničle, kar nastopa samo v primeru ko sta  $x_1=1$  in  $x_3=1$ .



<sup>1</sup><http://www.alldatasheet.com/view.jsp?Searchword=74HC139>

Rešitev 2. naloge:

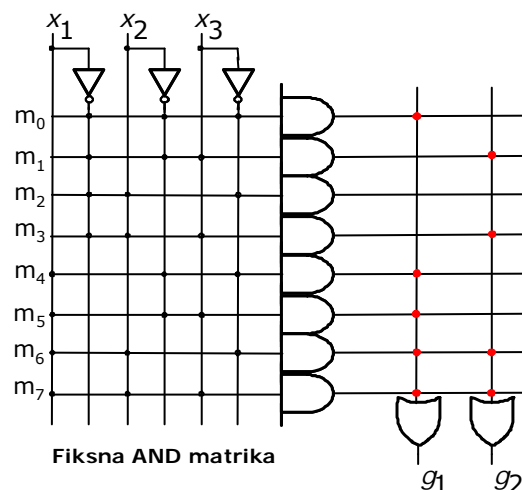
Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ev diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$\begin{aligned} g_1(x_1, x_2, x_3) &= x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= V(4, 6, 5, 7, 0) \end{aligned}$$

Podobno storimo še za preostale funkcije:

$$\begin{aligned} g_2 &= \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) \\ g_2(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\ g_2(x_1, x_2, x_3) &= V(1, 3, 6, 7) \end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ( $x_1 \ x_2 \ x_3$ ) od  $m_0$  do  $m_7$ . Številka minterma določa naslov lokacije ROM pomnilnika.



Narišemo celotno vezje ROM strukture in vstavimo pike (•) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

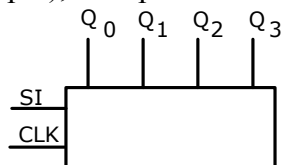
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

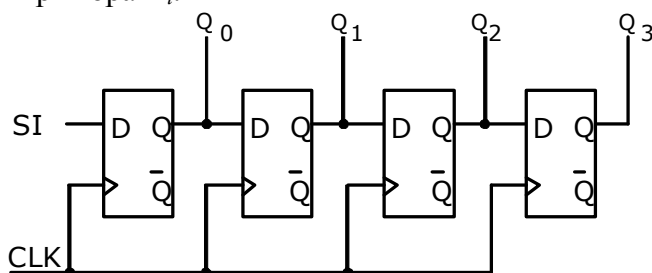


### Rešitev 3. naloge:

Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod  $SI$  (ang. serial input), in vzporedni izhod ( $Q_0, Q_1, Q_2, Q_3$ )



Zaporedno-vzporedni (SIPO) pomikalni register, realiziran s pomočjo D-FF, je veriga kaskadno vezanih D-FF, v kateri je izhod prejšnjega flip-flopa  $Q_{i-1}$  vezan na vhod naslednjega flip-flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz T-FF in 2/1 izbiralnikov, moramo pravzaprav realizirati celico D-FF s pomočjo T-FF in 2/1 izbiralnikov. V ta namen zapišemo tabelo D-FF, pri kateri dodamo izhodni stolpec  $T$  vhoda.

$D$	$Q(t)$	$Q(t+1)$	$T$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

XOR vrata moramo realizirati s pomočjo 2/1 izbiralnikov, zato zapišemo enačbo XOR funkcije:

$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

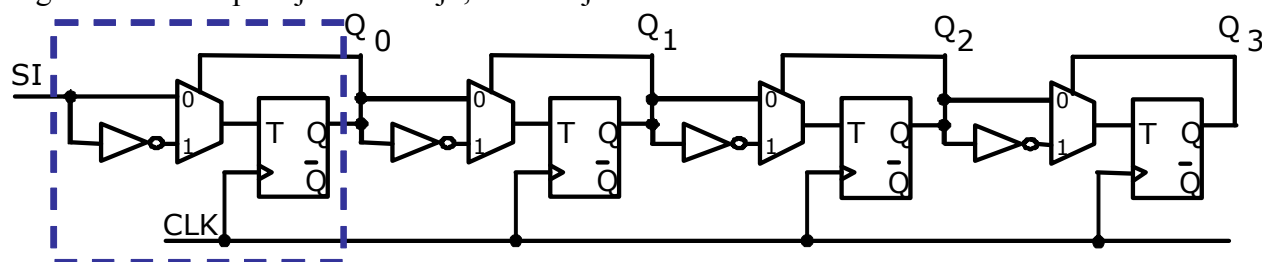
Iz tabele sledi, da je  $T$  vhod XOR operacija  $Q(t)$  in vhoda D-FF, ki ga realiziramo.

Funkcijo  $f$  realiziramo z izbiralnikom tako, da naredimo razvoj po spremenljivki  $x$  in dobimo:

$$Q(t+1) = Q(t) \oplus D$$

$x$	$f$
0	$y$
1	$y'$

Če nastali D-FF iz T-FF in 2/1 izbiralnika sestavimo skupaj v 4-bitni pomikalni register dobimo spodnjo realizacijo, v kateri je izvedba D-FF označena črtkano.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

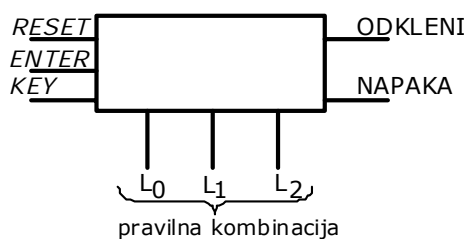
#### Rešitev 4. naloge:

Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev (*RESET*), ki postavlja ključavnico v začetno stanje, tipko (*ENTER*) za vnos nastavljene kombinacije in preklopnik (*KEY*) za vnos enega bita kombinacije. Izhod ključavnice je (*ODKLENI*), ki postane '1' ko uporabnik vnese pravilno kombinacijo in izhod (*NAPAKA*), ki postane '1' če je vnesena kombinacija napačna.

Uporabo ključavnice povzema spodnje zaporedje:

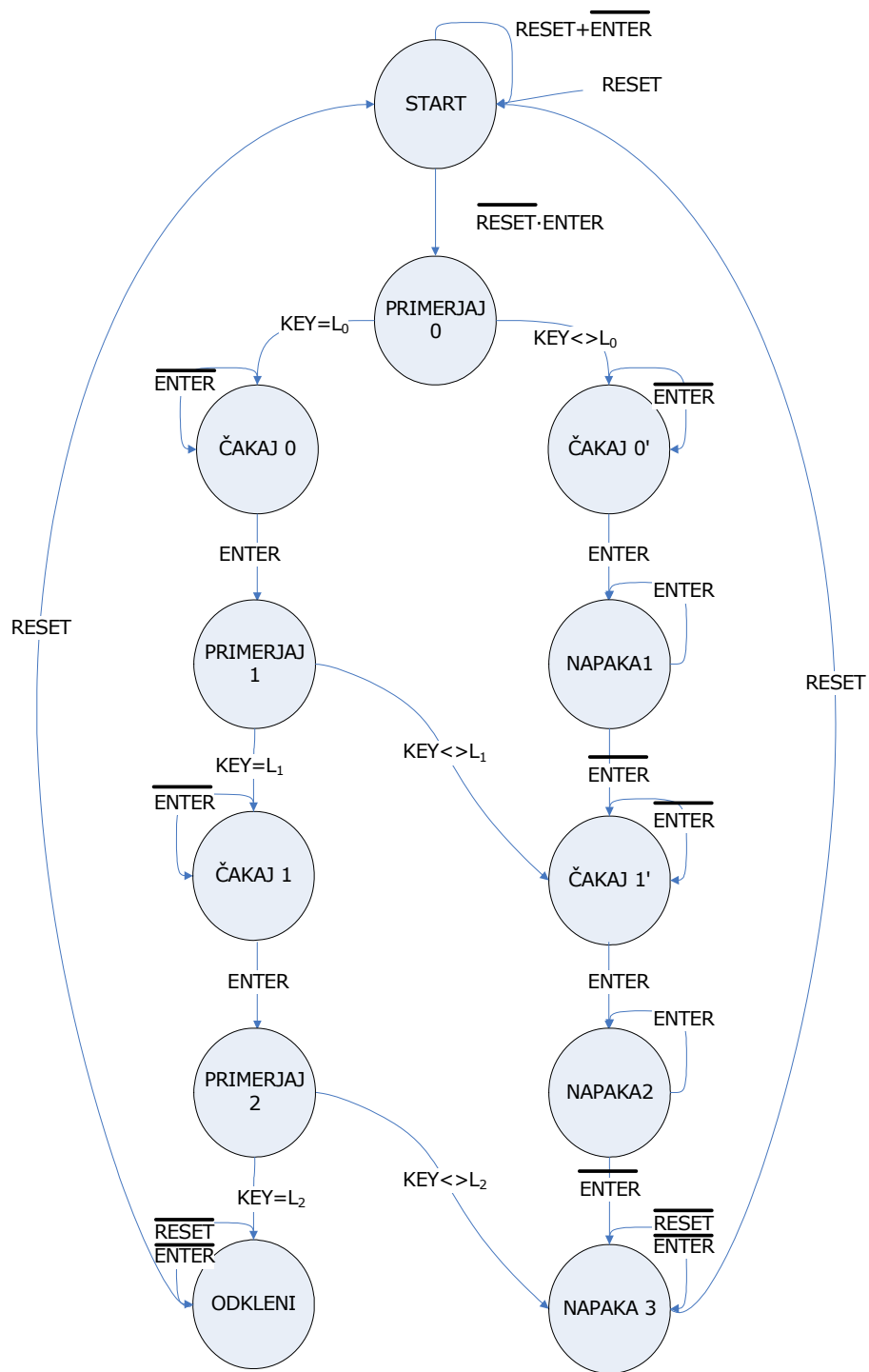
- 1.) pritisnemo *RESET*
- 2.) s preklopnikom *KEY* nastavimo bit kombinacije odklepanja
- 3.) pritisnemo *ENTER*
- 4.) izvede se primerjava i-tega bita ( $KEY = L_i$ )
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi *ODKLENI*='1', sicer se postavi *NAPAKA*='1'.
- 7.) ponoven pritisk na *RESET* nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti  $L_0$ ,  $L_1$  in  $L_2$ .



Ključavnica se ob vklopu ali ob ponastavitvi (*RESET*) nahaja v stanju *START*. V tem stanju uporabnik nastavi preklopnik *KEY* v stanje prvega bita kombinacije ('0' ali '1'). Iz tega stanja lahko pride v stanje *PRIMERJAJ 0* samo ob pogoju, da *RESET* ni pritisnjen in da je *ENTER* pritisnjen. V tem stanju je lahko vnesena kombinacija pravilna ( $KEY = L_0$ ) ali nepravilna ( $KEY \neq L_0$ ). Če je pravilna, potem preide v stanje *ČAKAJ 0*, v katerem čaka da uporabnik spusti tipko *ENTER*. Uporabnik v tem stanju nastavi drugi bit kombinacije (*KEY*) in ponovno pritisne *ENTER*. Avtomat preide v stanje *PRIMERJAJ 1*, od koder sta zopet dve možnosti. Omenjeni postopek se lahko ponavlja za več bitov kombinacije. Bistveno je, da pred vsako primerjavo postavimo stanje čakanja, v katerem čakamo, da uporabnik spusti tipko *ENTER*, nato nastavi bit kombinacije in šele nato preide v stanje nove primerjave ob ponovnem pritisku na *ENTER*. V zadnjem stanju primerjave avtomat preide v stanje *ODKLENI*.

Če se uporabnik pri vnašanju zmoti, preide avtomat v sekvenco stanj napake, po kateri mora vnesti še dve mesti kode (lahko samo dvakrat pritisne *ENTER*) in šele nato preide v stanje *NAPAKA 3*, v katerem se postavi izhod *NAPAKA*.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 18. 01. 2015

1. Realizirajte funkcijo  $f$  s čim manj izbiralniki 4/1.

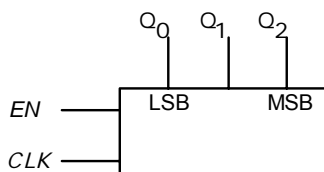
$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2 = x_1 \cdot x_2 \cdot x_3$
- $f_3$  = funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh vhodih.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko. Vezje PAL je narisano na hrbtni strani izpita.

3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor po Grayevi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2, Q_1, Q_0$ ), vhod za signal ure (CLK) in signal za omogočanje štetja (EN). Števec stoji, ko je EN='0' in šteje, ko je EN='1'. Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat. Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

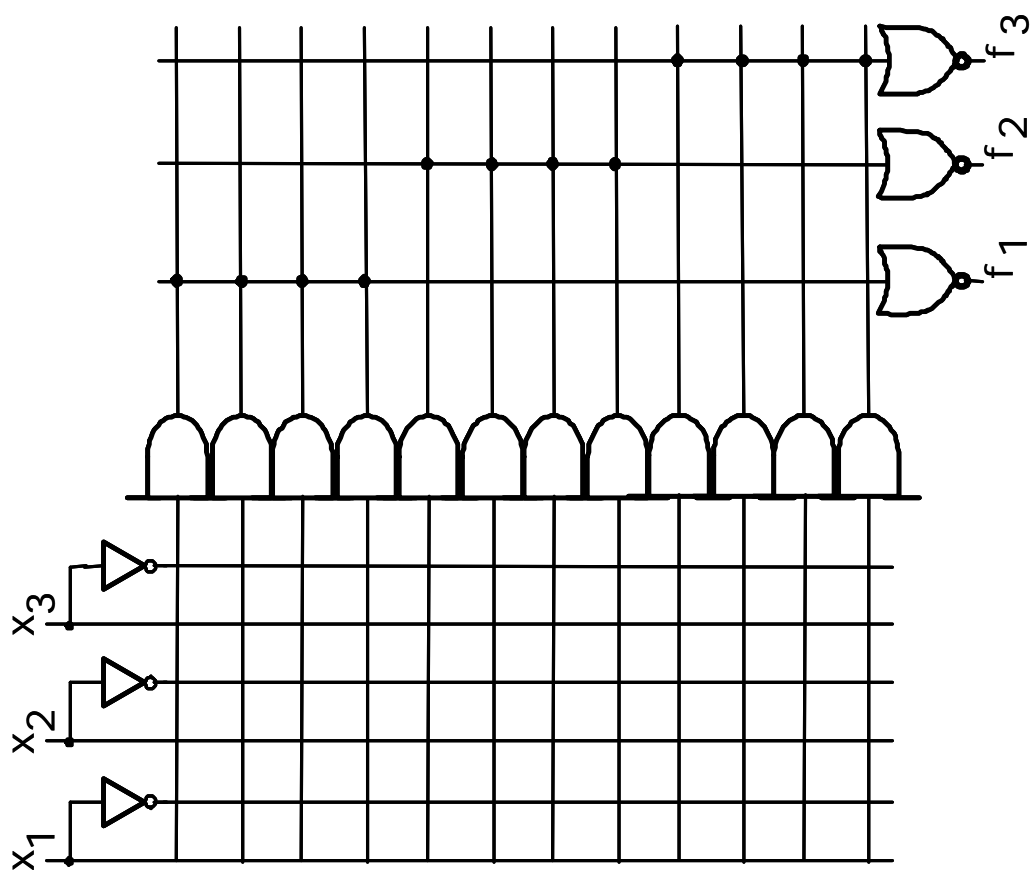
Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

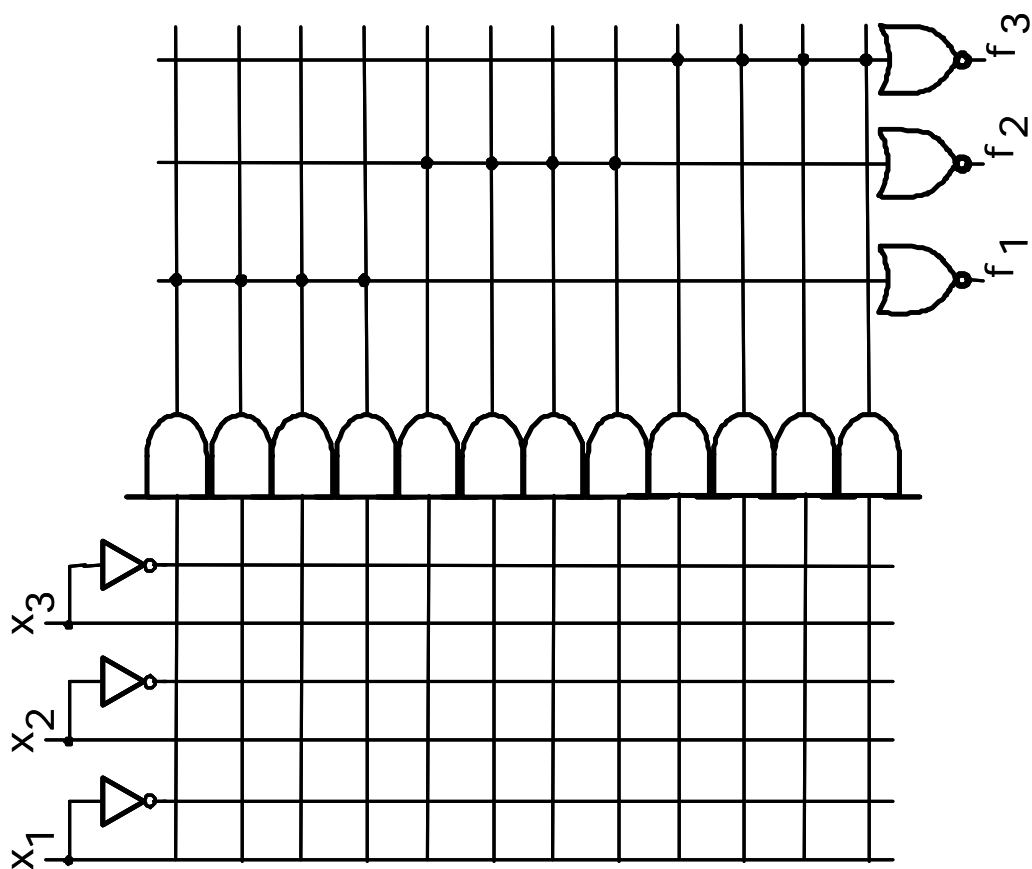
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.



Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

# DEVELOPMENT OF DIGITAL SYSTEMS

Examination

28. 1. 2015

1. Implement the function  $f$  using as few multiplexers 4/1 as possible.

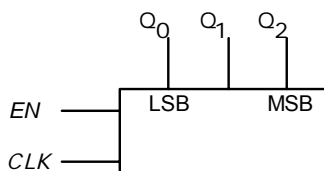
$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Implement following functions using an imaginary PAL3L3 circuit:

- $f_1 = x_1 \oplus x_2$
- $f_2 = x_1 \cdot x_2 \cdot x_3$
- $f_3 = \text{majority function of three variables}$   
( $f_3$  becomes '1', whenever majority of inputs is set to '1').

Imaginary circuit PAL3L3 is PAL-like structure with 3 inputs and 3 outputs. Each OR gate has 4 AND gates at its input. Designation L denotes inverted outputs. Draw the PAL3L3 structure and mark the programming points in the AND matrix.

3. Implement a three bit synchronous counter, which counts up in Gray code using T type flip-flops and any logical gates. The counter has a three bit count output ( $Q_2$ ,  $Q_1$ ,  $Q_0$ ), a clock input (CLK) and a count enable (EN) input. Counter holds state whenever (EN='0') and counts whenever (EN='1') Name the signals according to figure below.



4. Draw the state transition diagram of a Moore type finite state machine, which controls the movement of a garage door. The control circuit has an input DOOR and an input PROTECTION, which is set to '1' whenever a current limit of the motor is reached. The motor current limit input is used to detect both door end positions as well as for protection against obstacles in the door path. The control circuit has a two bit motor output:

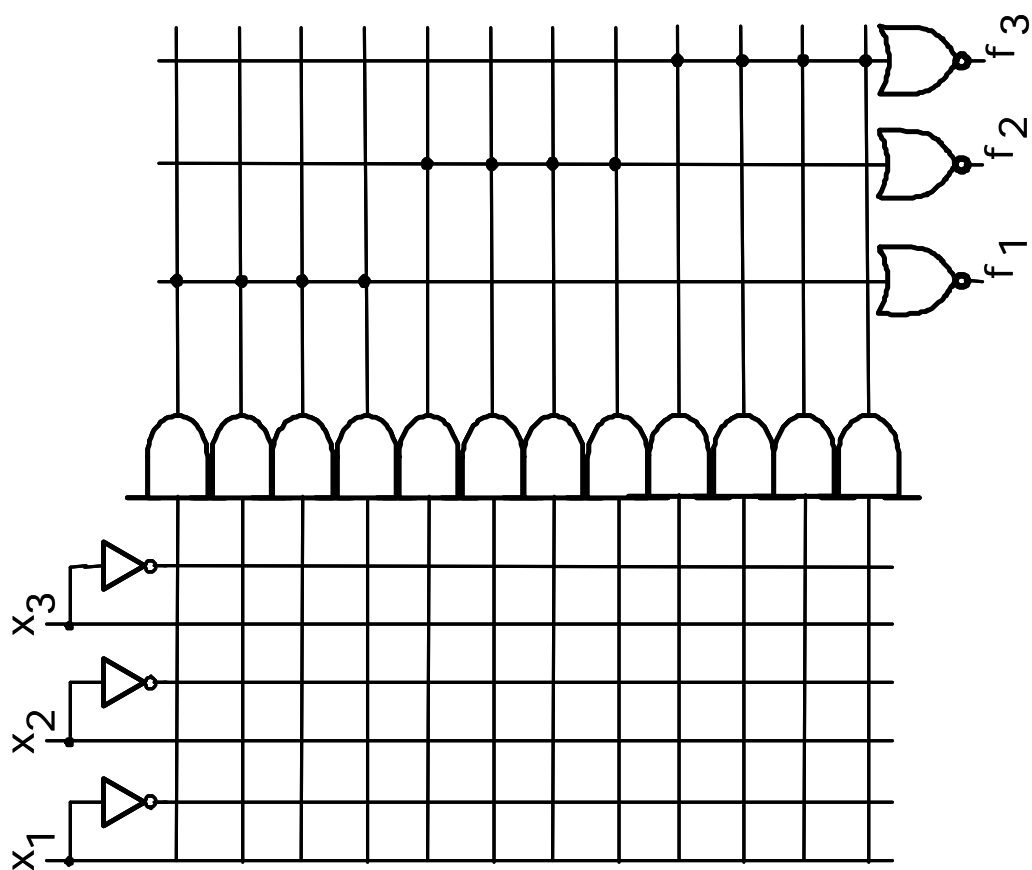
Operation code		Motor operation
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stop
0	1	door moving upwards
1	0	door moving downwards

When the door knob is pressed (DOOR='1'), the door starts to move upwards. If an obstacle is in the door path or the door reaches its upper end position (PROTECTION='1'), the motor stops. When the door knob is pressed again, the door starts to move downwards until the protection limit is reached again. After the door knob is pressed again, the process is repeated.

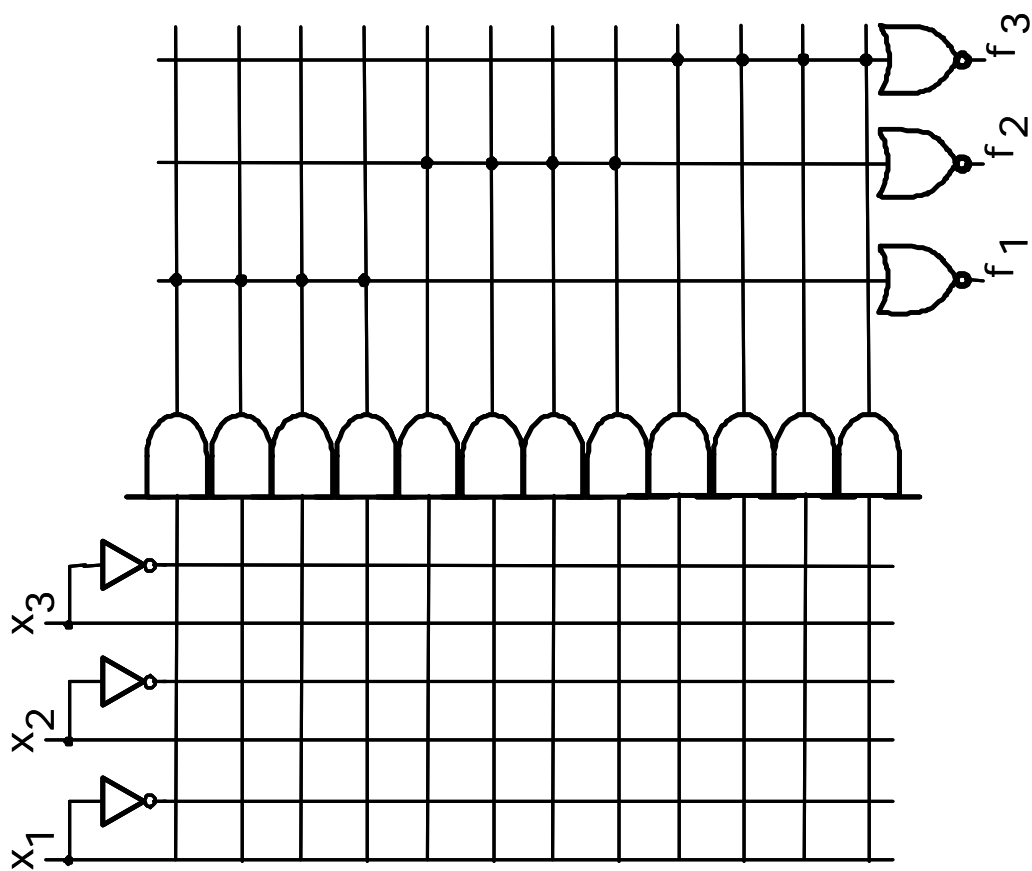
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.



Upon error, cross out the erroneous scheme and use the one given below!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

Rešitev 1. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

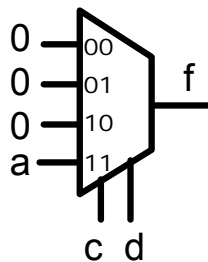
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

$$f_{PDNO}(a,b,c,d) = V(11,15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.



Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\begin{aligned} f_2 &= \overline{x_1 \cdot x_2 \cdot x_3} \\ \overline{f_2} &= \overline{x_1} + \overline{x_2} + \overline{x_3} \end{aligned}$$

Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

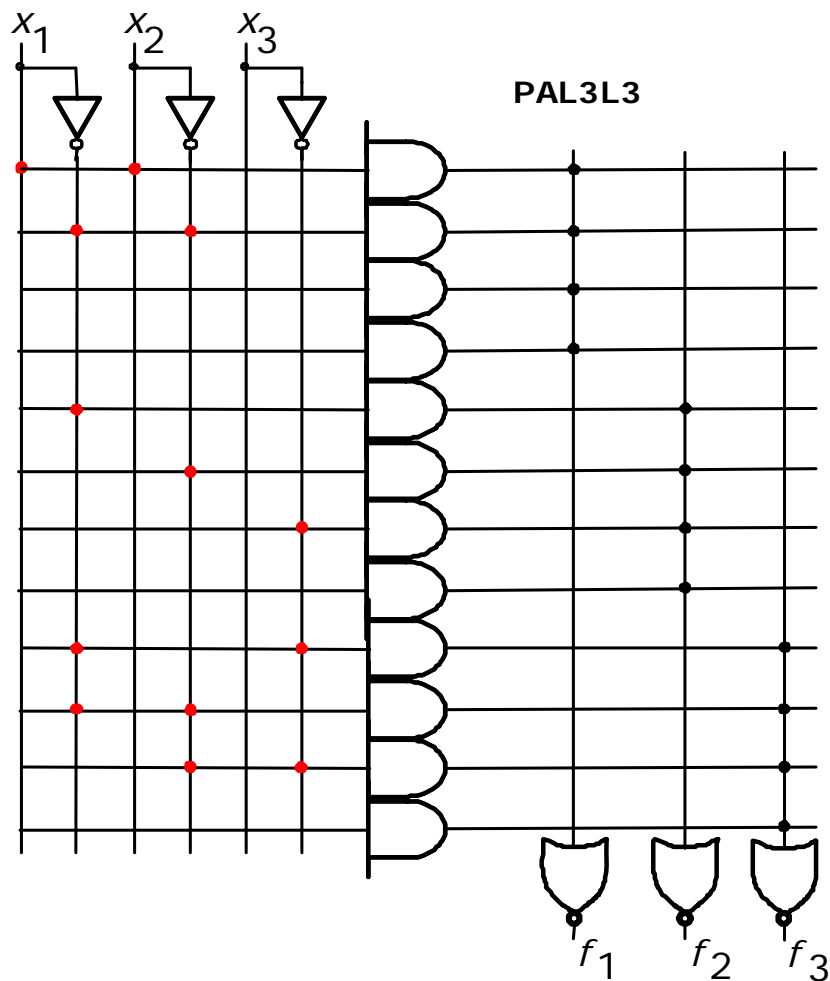
$$\overline{f_3} = \overline{x_1} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vezemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6–vhodna. Vezje PAL3L3 je AND–NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzgor po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
 ... 0, 1, 3, 2, 6, 7, 5, 4, 0...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		Q <sub>2</sub>	
T <sub>0</sub>	Q <sub>1</sub>	1	0
	Q <sub>0</sub>	0	1

$$T_0 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{Q_2} \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \cdot \overline{Q_0} + Q_1 \cdot Q_0) + Q_2 \cdot (\overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0})$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \oplus Q_0) + Q_2 \cdot (Q_1 \oplus Q_0)$$

Uvedemo novo spremenljivko x:

$$x = Q_1 \oplus Q_0$$

in jo vstavimo v izraz za T<sub>0</sub>:

$$T_0 = \overline{Q_2} \cdot \overline{x} + Q_2 \cdot x = \overline{Q_2} \oplus x$$

$$T_0 = 1 \oplus Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za T<sub>1</sub> narišemo Veitchev diagram:

		Q <sub>2</sub>	
T <sub>1</sub>	Q <sub>1</sub>	0	1
	Q <sub>0</sub>	0	1

Iz diagrama za T<sub>1</sub> sledi:

$$T_1 = Q_2 \cdot Q_1 \cdot Q_0 + \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0$$

$$T_1 = (Q_2 \cdot Q_1 + \overline{Q_2} \cdot \overline{Q_1}) \cdot Q_0$$

Operacija v oklepajih je ekvivalenca, zato enačbo lahko poenostavimo v:

$$T_1 = (\overline{Q_2} \oplus Q_1) \cdot Q_0$$

Podobno storimo še za T<sub>2</sub>:

		Q <sub>2</sub>	
T <sub>2</sub>	Q <sub>1</sub>	0	0
	Q <sub>0</sub>	1	0

Iz diagrama za T<sub>2</sub> sledi:

$$T_2 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR, zato enačbo lahko poenostavimo v:

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Manj potratno možnost realizacije predstavlja dvojiški 3-bitni sinhroni števec navzgor. Tak števec realiziramo s tremi T-FF in enimi AND vrati.

Nastalemu sinhronemu števcu na izhodu dodamo pretvornik kode iz dvojiškega v Gray-evo kodo z XOR vrati po enačbah:

$$G_{MSB} = B_{MSB}$$

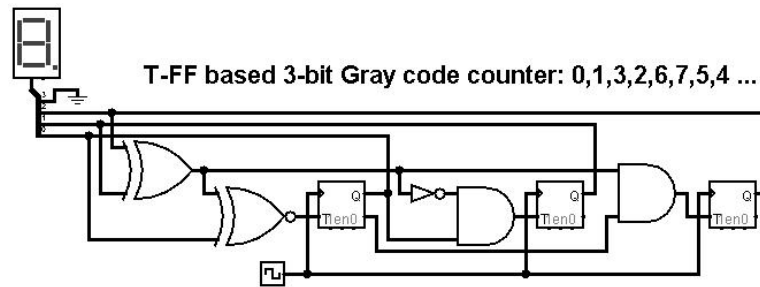
$$G_i = B_{i+1} \oplus B_i$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

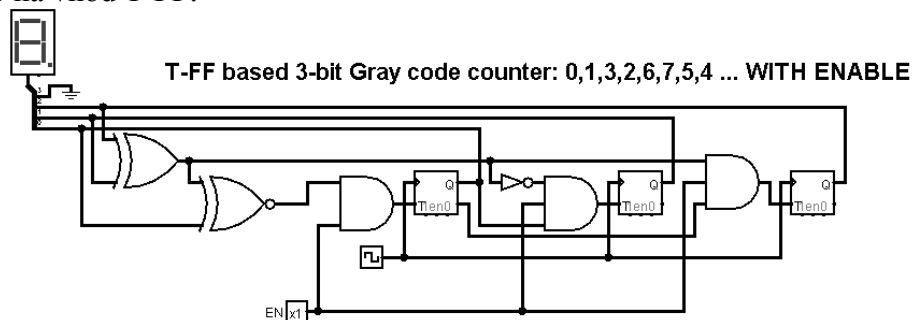
Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revB.circ



Funkcijo omogočanja štetja bi lahko realizirali s pisanjem tabele prehajanje stanj števca za štiri spremenljivke (ENABLE,  $Q_0$ ,  $Q_1$ ,  $Q_2$ ). Z malo razmišljanja se izognemo dolgotrajnemu pisanju: Števec bo obstal (ENABLE='0'), če so vsi vhodi T-FF enaki '0'. Števec bo štel (ENABLE='1'), ko bodo vhodi T-FF lahko spreminjali stanje, torej se bo izhod prejšnje stopnje nespremenjen pojavil na vhodu naslednje stopnje  $T_{i+1}$ . Povedano strnemo v tabelo za vhod vsakega T-FF:

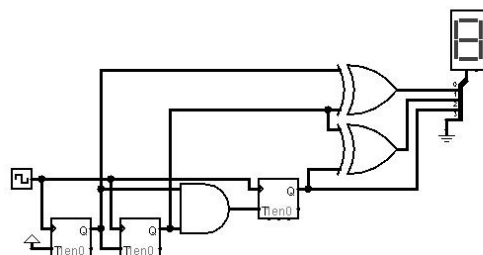
ENABLE	Izhod prejšnje stopnje	$T_{i+1}$
0	0	0
0	1	0
1	0	0
1	1	1

Funkcijo ENABLE torej realiziramo z dvovhodnimi AND vrati pred vhodom posameznega T-FF. Na vhoda AND vrat vodimo ENABLE in izhod iz prejšnje stopnje štetja, kot kaže desni del slike. Na LSB mestu štetja AND vrat ne potrebujemo, ampak ENABLE priključimo neposredno na vhod T-FF.



Enostavnejšo izvedbo štetja dosežemo z uporabo sinhronnega 3 bitnega dvojiškega števca in pretvornika kode iz dvojiške v Gray-evo kodo.

T-FF based 3-bit Gray code counter: 0,1,3,2,6,7,5,4 ...



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revC.circ

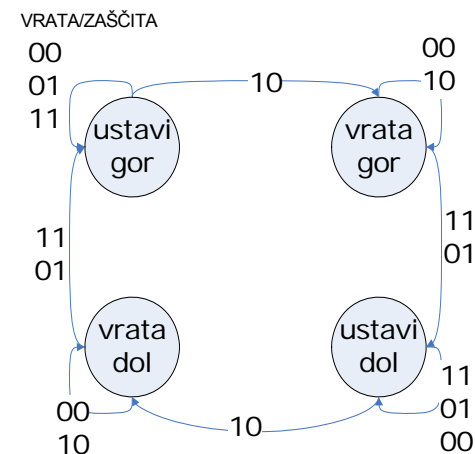
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



ime stanja	OP <sub>1</sub>	OP <sub>0</sub>
ustavi gor	0	0
vrata gor	0	1
vrata dol	1	0
ustavi dol	1	1

Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanja "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji "11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke

VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 11. 02. 2015

1. Določite popolno konjunktivno normalno obliko (PKNO) in popolno disjunktivno normalno obliko (PDNO) funkcije  $f$ .

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

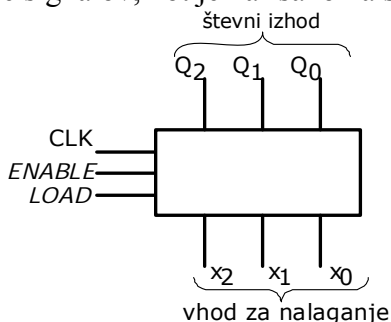
2. Realizirajte podano funkcijo  $f$  z redundancami s čim manj 4-bitnimi aritmetičnimi–logičnimi enotami (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Prikažite sintezo 3-bitnega sinhronega števca navzgor z omogočanjem štetja (*ENABLE*) in vzporednim nalaganjem (*LOAD*) z D flip–flopi, izbiralniki 2/1 in logičnimi vrati. Logika vseh krmilnih signalov je pozitivna.

S tovrstnim števcem realizirajte števec, ki šteje 2, 3, 4, 5, 2, 3, 4, 5 ...

Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Realizirajte Moore-ov avtomat končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z=1$  ko se na vhodu pojavi zaporedje "1001" ali "1111", sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno.

Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda.

w	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1
z	−	0	0	0	0	0	0	<b>1</b>	0	0	<b>1</b>	0	0	0	<b>1</b>	0	0	<b>1</b>

# DEVELOPMENT OF DIGITAL SYSTEMS

Examination

11. 2. 2015

1. Determine the complete product-of-sums form (PKNO) and the complete sum-of-products form (PDNO) of a given function  $f$ .

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

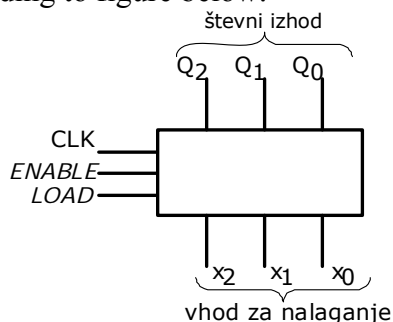
2. Implement a given function  $f$  using a single 4-bit arithmetic logic unit (ALU). Resulting negations of variables must be implemented within same ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Implement a synchronous 3 bit *up* counter with ENABLE and parallel LOAD function using D type flip-flops, multiplexers 2/1 and logic gates. Design the state transition table, determine the flip flop input equations and draw the resulting circuit using signal names as specified below. Control signals (ENABLE, LOAD) are positive logic.

Use a single designed counter with ENABLE and LOAD function to design a sequence counter, which will count: 2, 3, 4, 5, 2, 3, 4, 5 ...

Name the signals according to figure below.



4. Draw the state transition diagram for a Moore type finite state machine, which has an input  $w$  and output  $z$ . The output is set to '1' *whenever* an input sequence **1001** or **1111** is detected. Overlapping of sequences is allowed. Operation of finite state machine are summarized in the table below..

w	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1
z	—	0	0	0	0	0	0	<b>1</b>	0	0	<b>1</b>	0	0	0	<b>1</b>	0	0	<b>1</b>

## Rešitev 1. naloge

Funkcija je zapisana v večnivojski obliki, torej jo izrazimo v normalno obliko.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

Funkciji NOR ( $\downarrow$ ) in ekvivalence ( $\equiv$ ) izpišemo:

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 + x_2}) \cdot \overline{x_3} + \left( \overline{(\overline{x_2 \oplus x_4}) + x_1} \right)$$

Ekvivalenco smo izrazili kot negacijo XOR. Uporabimo De Morganov teorem:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2 \oplus x_4}) \cdot x_1$$

Izpišemo enačbo funkcije XOR ( $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$ ) in dobimo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2} \cdot \overline{x_4} + x_2 \cdot x_4) \cdot x_1$$

Razširimo še zadnjo konjunkcijo in rezultat je oblika MDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_4} \cdot x_1 + x_2 \cdot x_4 \cdot x_1$$

Če uporabimo lastnost Boole-ove algebre ( $\overline{\overline{a}} + a = 1$ ) lahko zapišemo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4}$$

Kar lahko zapišemo v obliki PDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4$$

$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$

PDNO pretvorimo v PKNO tako, da pregledamo manjkajoče minterme: 2, 3, 4, 5, 6, 7, 9, 11, 12, 14. Te minterme preslikamo preko tabele:

m <sub>i</sub>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M <sub>i</sub>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Funkcija v PKNO se torej glasi:

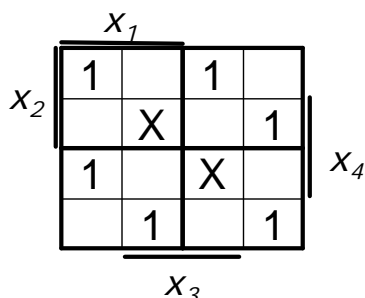
$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$

$$f_{PKNO}(x_1, x_2, x_3, x_4) = \&(13, 12, 11, 10, 9, 8, 6, 4, 3, 1)$$



## Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:



Funkcija vsebuje same diagonalne člene, zato realizacija v obliki KNO oz. DNO ne nudi minimalne oblike. Če se izkaže, da je funkcija linearna, jo lahko realiziramo s pomočjo XOR funkcij. Linearnost funkcije ugotovljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

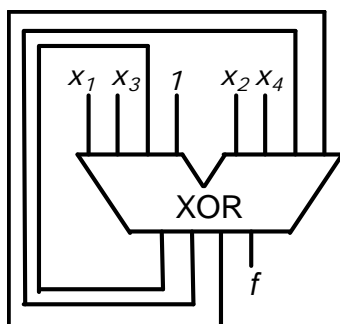
Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

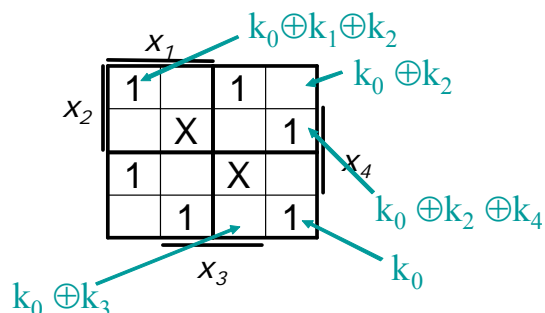
$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji uporabimo lastnost združevanja, ki velja za XOR funkcijo.

$$f(x_1, x_2, x_3, x_4) = 1 \oplus ((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$$



Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

### Rešitev 3. naloge:

trenutno stanje			naslednje stanje			D-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Podobno za D<sub>2</sub> narišemo Veitchev diagram

$D_2$ :

	$Q_2$			
$Q_1$	1	0	1	0
	1	1	0	0
	$Q_0$			

Za D<sub>2</sub> sledi:

$$D_2 = Q_2 Q_1' + Q_2 Q_0' + Q_2' Q_1 Q_0$$

Iz tabele prehajanja stanj števca določimo enačbe D-FF:

Za D<sub>0</sub> se iz tabele vidi  $D_0 = Q_0'$

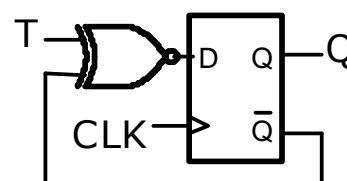
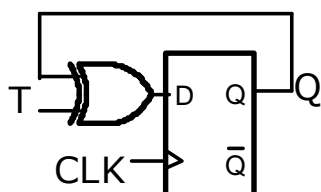
Za D<sub>1</sub> narišemo Veitchev diagram:

$D_1$ :

	$Q_2$			
$Q_1$	1	0	0	1
	0	1	1	0
	$Q_0$			

$$D_1 = Q_0 \oplus Q_1$$

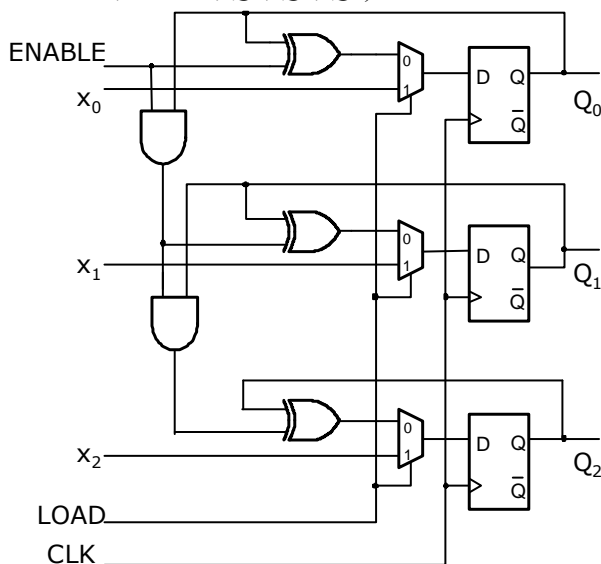
Če enačbo  $D_0 = Q_0'$  zapišemo kot  $D_0 = 1 \oplus Q_0$  in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:



Slika 1: Realizacija T-FF s pomočjo D-FF in XOR vrat (levo) in XNOR vrat (desno)

Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (*ENABLE*): Če prvemu "T-FF" (D-FF z XOR vrati) postavimo vhod  $T_0 = '0'$  namesto  $T_0 = '1'$ , vsi FF ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod  $T_0$  postavimo zunanji signal *ENABLE*, števec ne bo štel, ampak ohranjal stanje, če bo *ENABLE* = '0'. V verigi sinhronnega števca so namreč vsi FF vezani tako, da so odvisni od prvega FF. Če stanje ohranja prvi, ga bodo tudi vsi ostali.

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnega vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame za 5 spremenljivk (*ENABLE*, *LOAD*,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



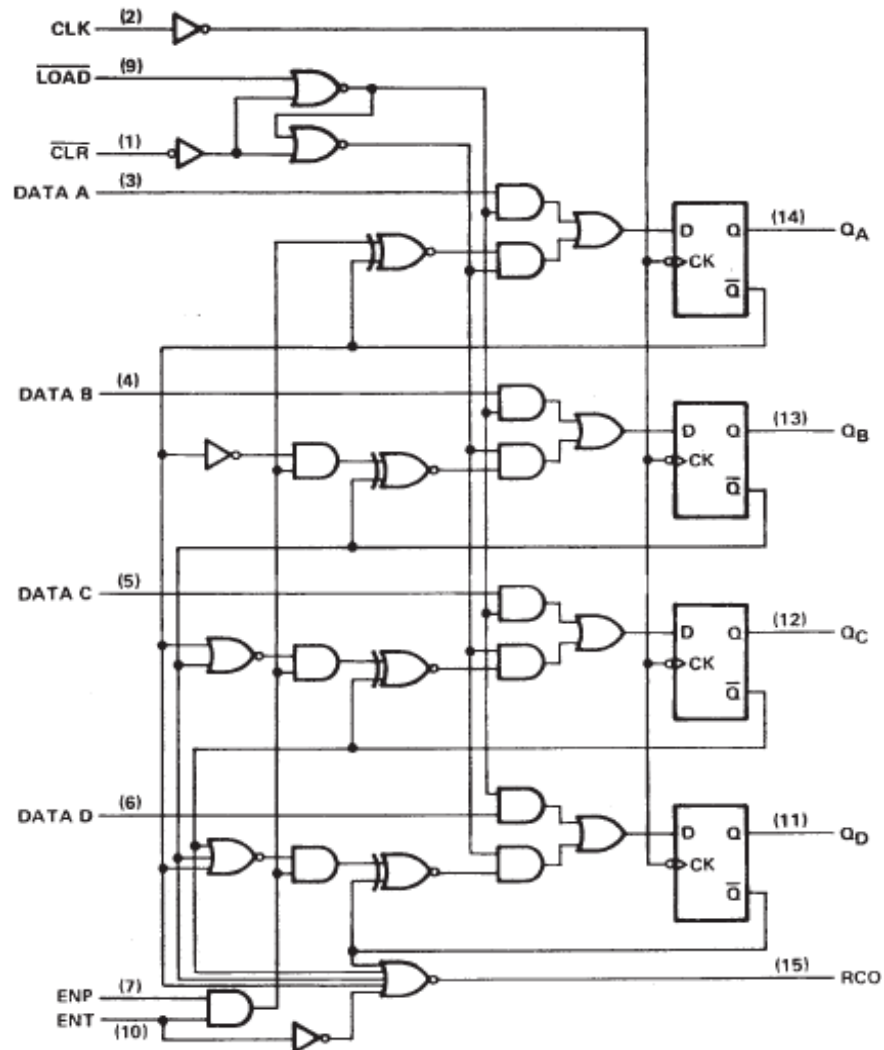
**Slika 2: Sinhroni števec z vzporednim nalaganjem (*LOAD*) in omogočanjem štetja (*ENABLE*) (3-bitna izvedba).**

Če želimo z nastalim števcem šteti naraščajoče 2, 3, 4, 5 ..., moramo ko števec pride do stanja 5 ( $Q_2Q_1Q_0=101_2$ ) števec postaviti nazaj na stanje v stanje 2 ( $Q_2Q_1Q_0=x_2x_1x_0=010_2$ ), torej signal *LOAD* dekodiramo s pomočjo 2–vhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se  $Q_2 = '1'$  in  $Q_0 = '1'$  v števeni sekvenci pojavlja samo enkrat – če bi se večkrat, bi morali dekodirati tudi  $Q_1$ . Pri tovrstnih števcih ponavadi uporabljamo še en signal *RESET*, s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod D–FF za brisanje asinhrono priključimo signal *RESET*.

Nastali strukturi števca bi lahko na isti način dodali še četrti bit. Tako bi dobili podobno strukturo kot je 4-bitni sinhroni števec z vzporednim nalaganjem 74163<sup>1</sup>. Pri spodnji realizaciji tega vezja so uporabljeni D–FF, proženi na negativni rob signala ure (*CLK*). Štetje omogočimo s signaloma *ENP*, *ENT* (ang. enable parallel, enable transfer). Štetje je izvedeno tako, da se D–FF spremenijo v T–FF, kar dosežemo s pomočjo XNOR vrat, ki imajo en vhod vezan na števeni signal ( $ENT=ENP='1'$ ), drug vhod pa na izhod  $Q'$  FF. AND vrata pred XNOR zagotavljajo prenehanje štetja, če velja  $ENT \cdot ENP \neq '1'$ . Na vhodu D–FF je vezan enostaven izbiralnik iz dveh AND vrat, vezanih na OR vrata. Ta izbiralnik določa, ali bo števec štel, ali bo vzporedno naložil vrednost. Zgornja AND vrata izbiralnika so krmiljena s signalom  $LOAD \cdot CLR'$ , spodnja pa z  $LOAD' \cdot CLR'$ . Čim velja, da je  $CLR' = '1'$  in  $LOAD' = '0'$ , se v D–FF asinhrono naloži vsebina na vhodih  $Q_D$   $Q_C$   $Q_B$   $Q_A = DATA_D$   $DATA_C$   $DATA_B$   $DATA_A$ , medtem ko dokler velja  $LOAD' = '1'$  in  $CLR' = '1'$ , bo števec štel navzgor. Če je  $CLR' = '0'$ , je na obeh vhodih OR vrat izbiralnika '0' in stanje vseh D–FF se asinhrono postavi na  $Q_DQ_CQ_BQ_A = '0000'$ . Števeni izhodi so  $Q_D$   $Q_C$   $Q_B$   $Q_A$ . Izhod *RCO* (*ripple carry out*) se postavi na '1', ko je števec v stanju "1111" ob tem, da

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74163>

je  $ENT='1'$ . Signal RCO je izveden z NOR vrati na vhode katerih so priključeni negirani izhodi D-FF, kar je po De Morgan-ovem teoremu ekvivalent AND vratom.



Slika 3: Struktura 4-bitnega MSI sinhronega števca z vzporednim nalaganjem (74163).

#### Rešitev 4. naloge:

Prikazali bomo izvedbo avtomata končnih stanj Moore-ove izvedbe.

Diagram prehajanja stanj:

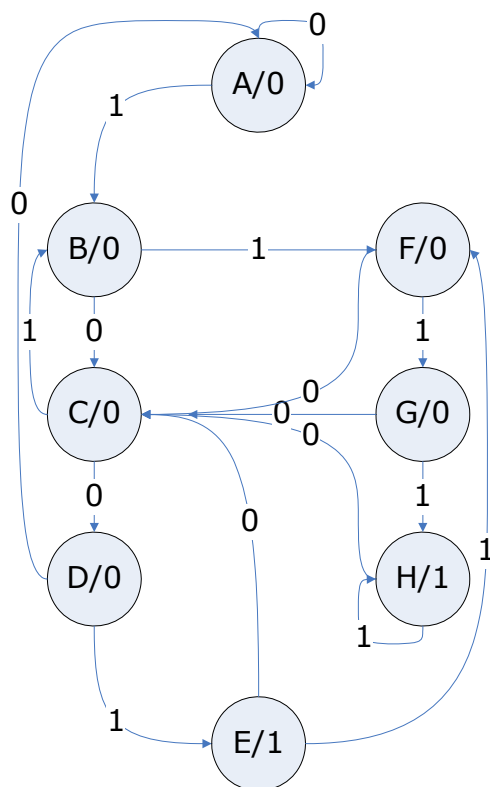


Diagram stanj začnemo risati tako, da najprej narišemo začetno stanje (A) v katerega se vračamo vedno, kadar sekvenca ne bo podobna tisti, ki jo zaznavamo (1111 oz. 1001). V tem stanju vztrajamo toliko časa, dokler je na vhodu '0', saj se nobena od sekvenc ne začneja z '0'. Ko pride na vhod prva '1', preidemo v drugo stanje (B), v katerem imamo dve možnosti, saj se sekvenci razlikujeta na drugem mestu: Če na vhod tega stanja pride '0', bomo zaznavali sekvence tipa '10XX', če pa pride '1', potem pa sekvence tipa '11XX'. Recimo, da v stanju B na vhod pride '0', torej napredujemo v stanje C. V tem stanju se ponovno lahko pojavi '0' – torej bi bila na vhodu sekvenca

tipa '100X', kar bi nas vodilo do naslednjega stanja D. Če pa se pojavi na vhodu logična '1' je sicer sekvenca napačna, vendar moramo to predstaviti tako kot da je na vhod prišla že prva '1' od sekvence - naloga namreč pravi, da se sekvence lahko med seboj prekrivajo. S podobnim načinom razmišljanja narišemo naslednje stanje E, v katero napredujemo iz D samo, ko vanj pride '1', kar pomeni da smo v tem stanju zaznali sekvenco "1001", zato je v tem stanju izhod vezja enak '1'. Stanja F, G in H služijo pomnjenju koliko enic je prišlo na vhod vezja in sicer: stanje F pomenita "11" na vhodu vezja, stanje G tri enice in zadnje stanje H četrto enico sekvence. Slednje stanje vztraja, dokler je na vhodu '1', saj tako rešimo prekrivanje vzorca '1111'. Če pa v kateremkoli od stanj F, G in H pride na vhod '0', se postavimo v stanje C, saj to stanje pomeni, da je na vhodu sekvenca '10XX'.

Tabela prehajanja stanj:

trenutno stanje		naslednje stanje		izhod
pomen stanja	koda stanja	w=0	w=1	z
začetno stanje	A	A	B	0
prva enica '1001' ali '1111' na vhodu	B	C	F	0
prva ničla '1001' na vhodu	C	D	B	0
druga ničla '1001' na vhodu	D	A	E	0
druga enica '1001' na vhodu	E	C	F	1
druga enica '1111' na vhodu	F	C	G	0
tretja enica '1111' na vhodu	G	C	H	0
četrtta enica '1111' na vhodu	H	C	H	1

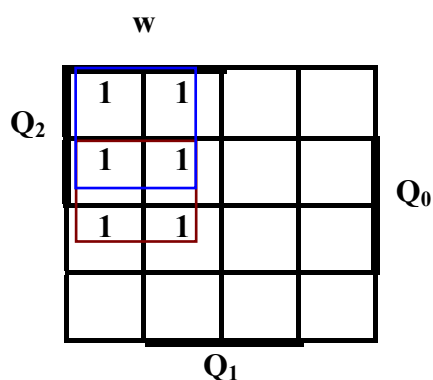
Za izvedbo bomo rabili najmanj 3 FF, saj je stanj 8. Izberemo zaporedno kodiranje stanj se pravi A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.

Vzbujalna tabela:

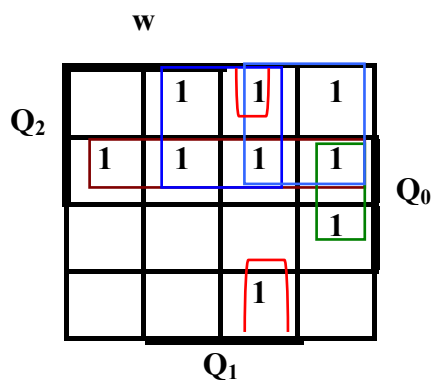
trenutno stanje				naslednje stanje						izhod
				w=0			w=1			
	Q <sub>2</sub> (t)	Q <sub>1</sub> (t)	Q <sub>0</sub> (t)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	z
A	0	0	0	0	0	0	0	0	1	0
B	0	0	1	0	1	0	1	0	1	0
C	0	1	0	0	1	1	1	0	1	0
D	0	1	1	0	0	0	1	0	0	0
E	1	0	0	0	1	0	1	0	1	1
F	1	0	1	0	1	0	1	1	0	0
G	1	1	0	0	1	0	1	1	1	0
H	1	1	1	0	1	0	1	1	1	1

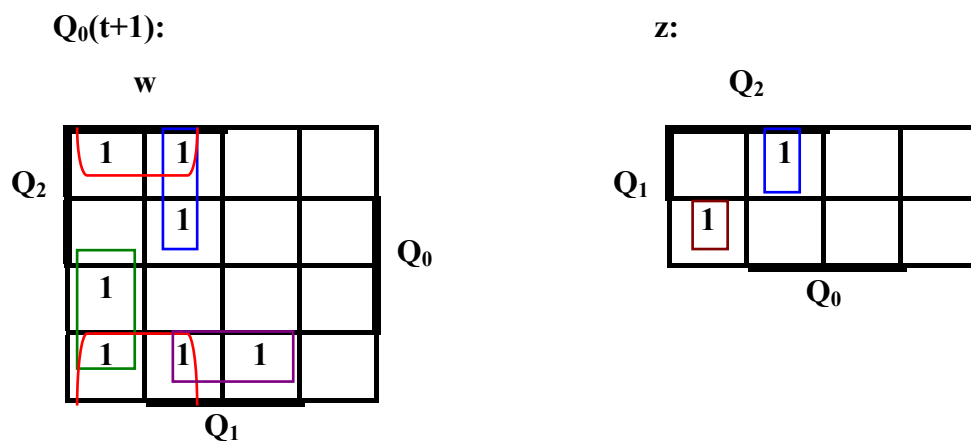
Iz vzbujalne tabele sestavimo tri Veitcheve diagrame:

$Q_2(t+1)$ :



$Q_1(t+1)$ :





Funkcije v Veitchevih diagramih minimiziramo in dobimo enačbe za realizacijo s pomočjo D flip-flopov. Realizacija s pomočjo D flip-flopov je najbolj enostavna, saj je enačba D flip-flopa:

$$D = Q(t+1) \quad (0.1)$$

kar pomeni, da lahko iz minimizacije Veitchevih diagramov naslednjih stanj zapišemo enačbe za vhode D flip-flopov:

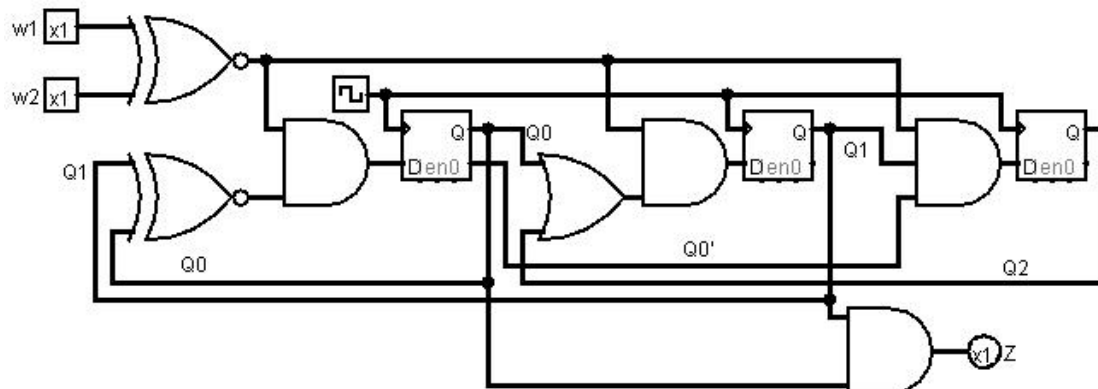
$$\begin{aligned} D_2 &= Q_2(t+1) = w \cdot Q_2(t) + \bar{w} \cdot Q_0(t) = w \cdot (Q_2(t) + Q_0(t)) \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + \bar{Q}_1(t) \cdot Q_0(t) + Q_1(t) \cdot \bar{Q}_0(t)) = \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + Q_1(t) \oplus Q_0(t)) \\ D_0 &= Q_0(t+1) = w \cdot \bar{Q}_0(t) + w \cdot \bar{Q}_2(t) \cdot \bar{Q}_1(t) + w \cdot Q_2(t) \cdot Q_1(t) + \bar{Q}_1(t) \cdot \bar{Q}_0(t) = \\ D_0 &= Q_0(t+1) = w \cdot (\bar{Q}_0(t) + \bar{Q}_2(t) \oplus \bar{Q}_1(t)) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t) \end{aligned} \quad (0.2)$$

Izhod z lahko zapišemo kot:

$$z = Q_2(t) \cdot (Q_1(t) \cdot Q_0(t) + \bar{Q}_1(t) \cdot \bar{Q}_0(t)) = Q_2(t) \cdot (Q_1(t) \oplus Q_0(t)) \quad (0.3)$$

Vezje avtomata narišemo iz enačb (0.2) in (0.3).

Opis delovanja in vezje avtomata, ki primerja enakost znotraj 4 period signala ure je v predlogah vaj na domači strani predmeta v imeniku Logisim\fsm\fsm\_four\_periodes\_of\_equality.circ



# RAZVOJ DIGITALNIH SISTEMOV

Izpit 29. 01. 2016

1. Realizirajte funkcijo  $f$  s čim manj izbiralniki 4/1.

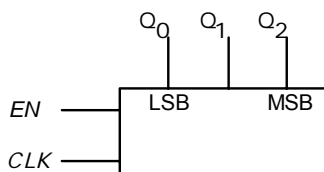
$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2 = x_1 \cdot x_2 \cdot x_3$
- $f_3$  = funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh vseh.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko. Vezje PAL je narisano na hrbtni strani izpita.

3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor po Grayevi kodi s T flip-flopi in logičnimi vrati. Števec ima 3-bitni števeni izhod ( $Q_2, Q_1, Q_0$ ), vhod za signal ure (CLK) in signal za omogočanje štetja (EN). Števec stoji, ko je EN='0' in šteje, ko je EN='1'. Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat. Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

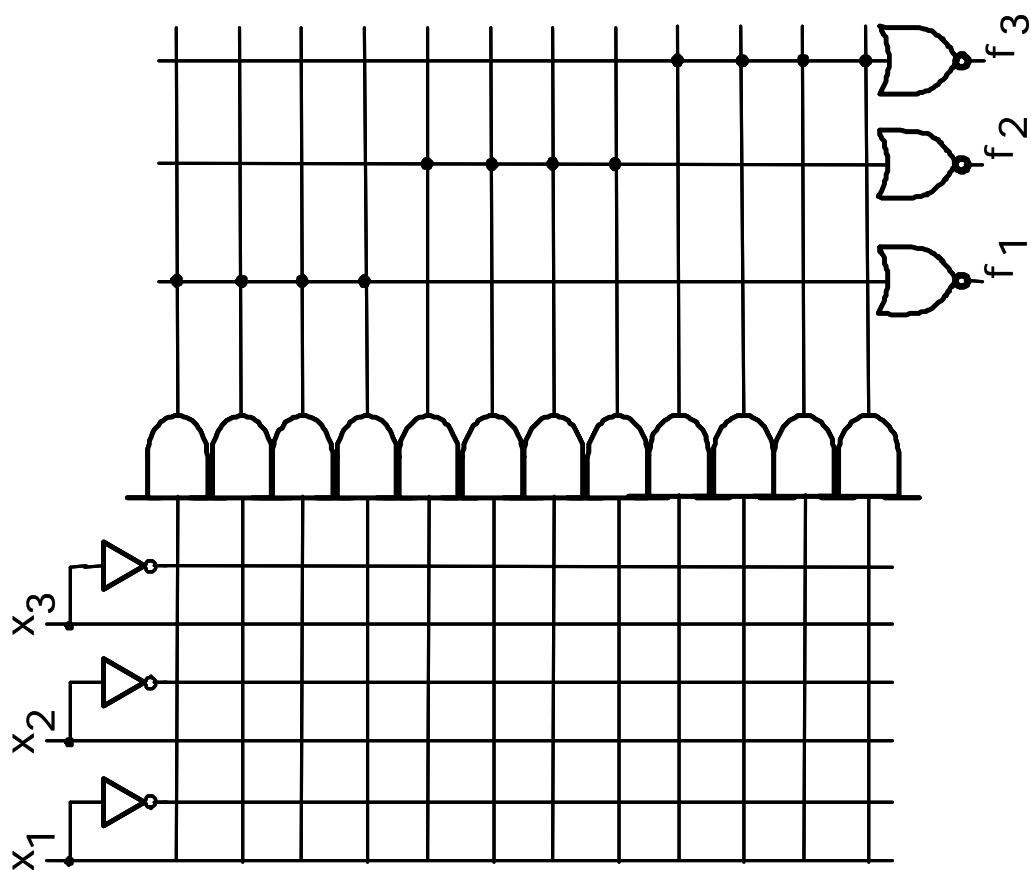
Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

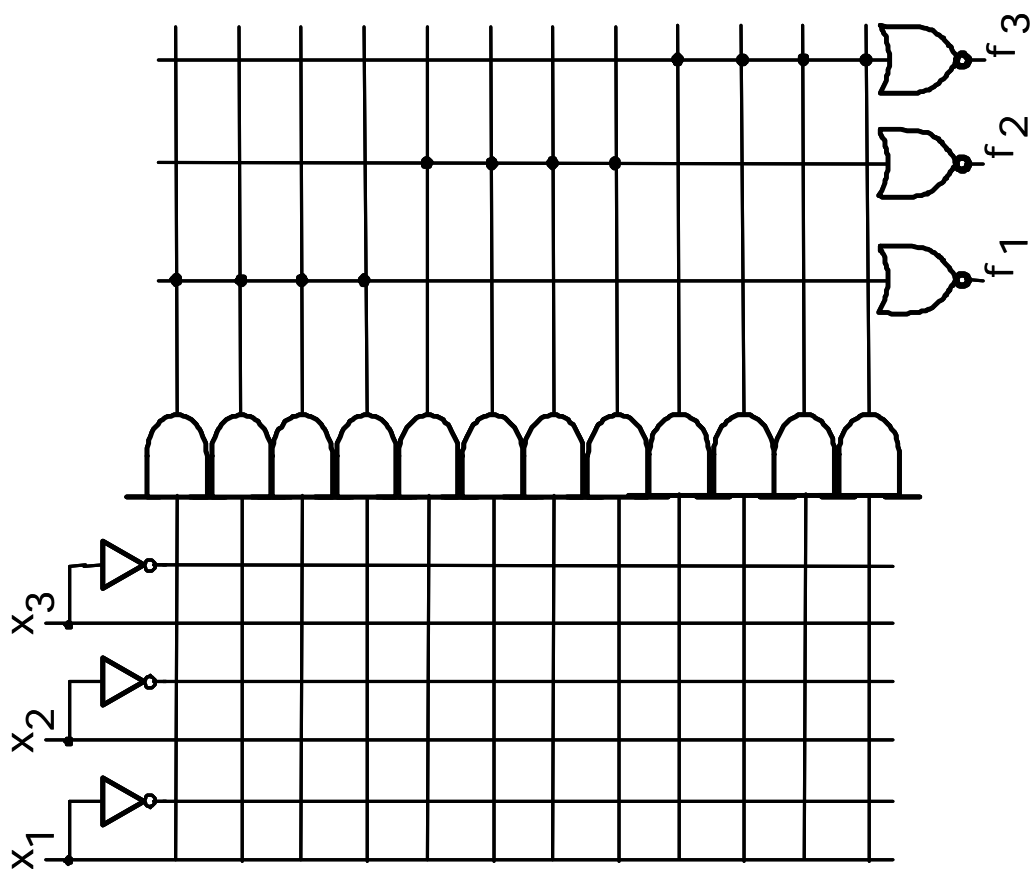
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.





Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

## Rešitev 1. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

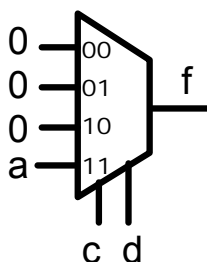
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

$$f_{PDNO}(a,b,c,d) = V(11,15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

## Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\overline{f_2} = \overline{x_1 \cdot x_2 \cdot x_3}$$

$$\overline{f_2} = \overline{x_1} + \overline{x_2} + \overline{x_3}$$

Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

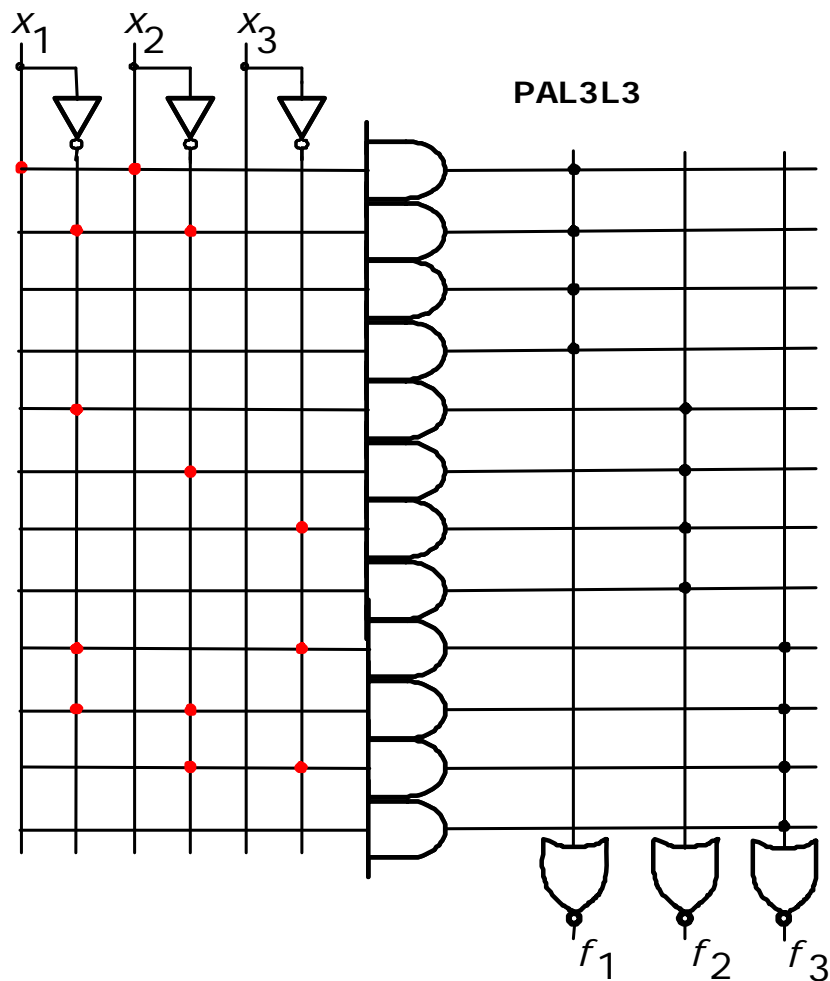
$$\overline{f_3} = \overline{x_1 \cdot x_3} + \overline{x_1 \cdot x_2} + \overline{x_2 \cdot x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vezemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6–vhodna. Vezje PAL3L3 je AND–NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca navzgor po Gray-evi kodi. Desetiška števna sekvenca po 3-bitni Grayevi kodi se glasi:  
 ... 0, 1, 3, 2, 6, 7, 5, 4, 0...

Števno sekvenco zapišemo v tabelo:

trenutno stanje			naslednje stanje			enačbe T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> narišemo Veitchev diagram. Funkcija je funkcija linearna, zato jo bomo izrazili z XOR operacijami.

		Q <sub>2</sub>	
T <sub>0</sub>	Q <sub>1</sub>	1	0
	Q <sub>0</sub>	0	1

$$T_0 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{Q_2} \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \cdot \overline{Q_0} + Q_1 \cdot Q_0) + Q_2 \cdot (\overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0})$$

$$T_0 = \overline{Q_2} \cdot (\overline{Q_1} \oplus Q_0) + Q_2 \cdot (Q_1 \oplus Q_0)$$

Uvedemo novo spremenljivko x:

$$x = Q_1 \oplus Q_0$$

in jo vstavimo v izraz za T<sub>0</sub>:

$$T_0 = \overline{Q_2} \cdot \overline{x} + Q_2 \cdot x = \overline{Q_2} \oplus x$$

$$T_0 = 1 \oplus Q_2 \oplus Q_1 \oplus Q_0$$

Podobno za T<sub>1</sub> narišemo Veitchev diagram:

		Q <sub>2</sub>	
T <sub>1</sub>	Q <sub>1</sub>	0	1
	Q <sub>0</sub>	0	1

Iz diagrama za T<sub>1</sub> sledi:

$$T_1 = Q_2 \cdot Q_1 \cdot Q_0 + \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0$$

$$T_1 = (Q_2 \cdot Q_1 + \overline{Q_2} \cdot \overline{Q_1}) \cdot Q_0$$

Operacija v oklepajih je ekvivalenca, zato enačbo lahko poenostavimo v:

$$T_1 = (\overline{Q_2} \oplus Q_1) \cdot Q_0$$

Podobno storimo še za T<sub>2</sub>:

		Q <sub>2</sub>	
T <sub>2</sub>	Q <sub>1</sub>	0	0
	Q <sub>0</sub>	1	0

Iz diagrama za T<sub>2</sub> sledi:

$$T_2 = \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$$

$$T_2 = (\overline{Q_2} \cdot Q_1 + Q_2 \cdot \overline{Q_1}) \cdot \overline{Q_0}$$

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Operacija v oklepajih je negacija XOR, zato enačbo lahko poenostavimo v:

$$T_2 = (Q_2 \oplus Q_1) \cdot \overline{Q_0}$$

Manj potratno možnost realizacije predstavlja dvojiški 3-bitni sinhroni števec navzgor. Tak števec realiziramo s tremi T-FF in enim AND vrati.

Nastalemu sinhronemu števcu na izhodu dodamo pretvornik kode iz dvojiškega v Gray-evo kodo z XOR vrati po enačbah:

$$G_{MSB} = B_{MSB}$$

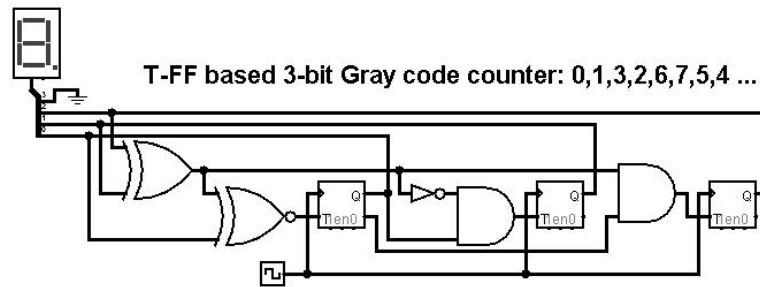
$$G_i = B_{i+1} \oplus B_i$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

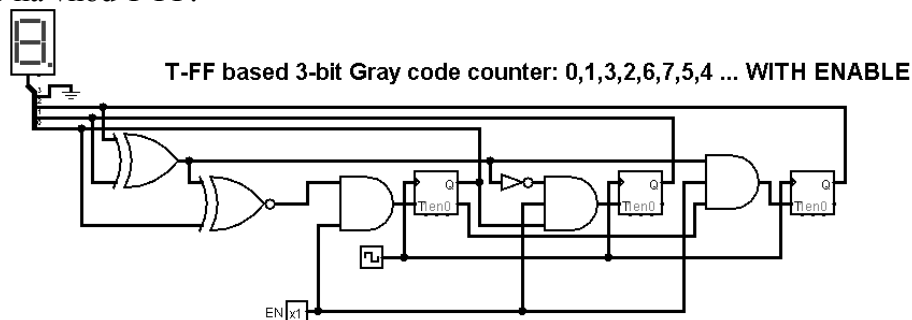
Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revB.circ



Funkcijo omogočanja štetja bi lahko realizirali s pisanjem tabele prehajanje stanj števca za štiri spremenljivke (ENABLE,  $Q_0$ ,  $Q_1$ ,  $Q_2$ ). Z malo razmišljanja se izognemo dolgotrajnemu pisanju: Števec bo obstal (ENABLE='0'), če so vsi vhodi T-FF enaki '0'. Števec bo štel (ENABLE='1'), ko bodo vhodi T-FF lahko spreminjali stanje, torej se bo izhod prejšnje stopnje nespremenjen pojavil na vhodu naslednje stopnje  $T_{i+1}$ . Povedano strnemo v tabelo za vhod vsakega T-FF:

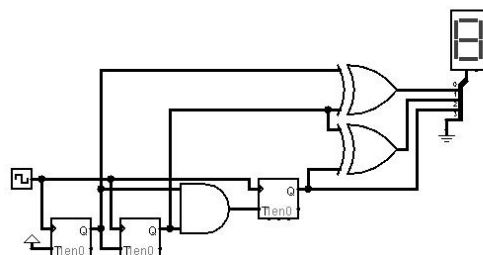
ENABLE	Izhod prejšnje stopnje	$T_{i+1}$
0	0	0
0	1	0
1	0	0
1	1	1

Funkcijo ENABLE torej realiziramo z dvovhodnimi AND vrati pred vhodom posameznega T-FF. Na vhoda AND vrat vodimo ENABLE in izhod iz prejšnje stopnje štetja, kot kaže desni del slike. Na LSB mestu štetja AND vrat ne potrebujemo, ampak ENABLE priključimo neposredno na vhod T-FF.



Enostavnejšo izvedbo štetja dosežemo z uporabo sinhronnega 3 bitnega dvojiškega števca in pretvornika kode iz dvojiške v Gray-evo kodo.

T-FF based 3-bit Gray code counter: 0,1,3,2,6,7,5,4 ...



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\counter\3-bit Gray code counter\_revC.circ

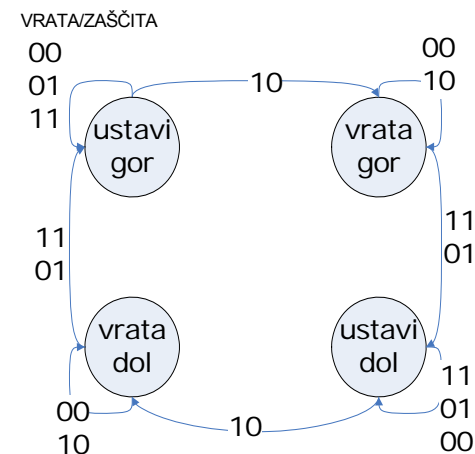
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



ime stanja	OP <sub>1</sub>	OP <sub>0</sub>
ustavi gor	0	0
vrata gor	0	1
vrata dol	1	0
ustavi dol	1	1

Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanja "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji "11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke

VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

06. 02. 2017

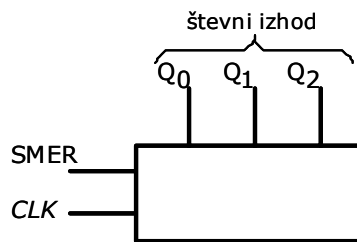
1. Realizirajte funkcijo  $f$  z dvovhodnimi vpoglednimi tabelami LUT2 (ang. look-up table).

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

2. Realizirajte podano funkcijo  $f$  z eno 4-bitno aritmetično-logično enoto (ALU). Morebitne negacije vhodnih spremenljivk izvedite z ALU.

$$f(a, b, c, d, e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

3. Prikažite sintezo sinhronega dvosmernega 3-bitnega dvojiškega števca z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov. Števec ima vhod SMER, ki določa smer štetja in izhod štetja  $Q_2$ ,  $Q_1$ ,  $Q_0$ . Če je SMER='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Minimizirajte podani avtomat končnih stanj z uporabo metode z razdelki ter zapišite tabelo prehajanja stanj nastalega minimalnega avtomata.

<i>Trenutno stanje</i>	<i>Naslednje stanje</i>		<i>Izhod</i>
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



# DEVELOPMENT OF DIGITAL SYSTEMS

Written examination

06. 02. 2017

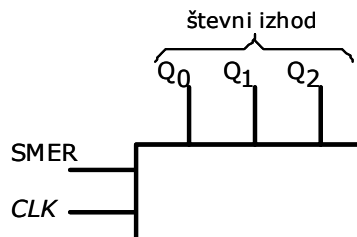
1. Implement a given function  $f$  using two input look-up-tables (LUT2).

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

2. Implement a given function  $f$  using a single 4-bit arithmetic logic unit (ALU). Any resulting negations of variables must be implemented within this ALU.

$$f(a, b, c, d, e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

3. Implement a synchronous 3 bit *up/down* binary counter using T type flip-flops and logic gates: Design the state transition table, determine the flip-flop input equations and draw the resulting circuit using signal names as specified below. The counter has a count direction input (SMER) and a 3-bit binary output  $Q_2, Q_1, Q_0$ . If (SMER='0'), the counter will count upwards, otherwise downwards. Name the signals according to figure below.



4. Minimize a given Moore type finite state machine.

<i>Current state</i>	<i>Next state</i>		<i>Output</i>
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

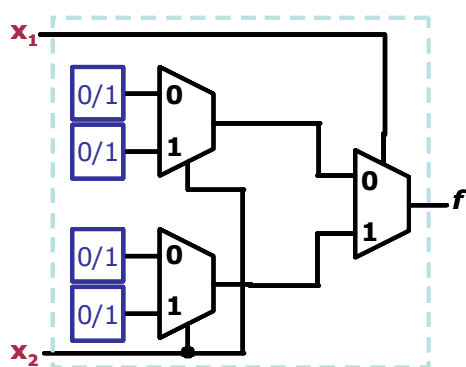
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

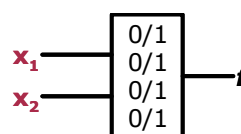
Funkcija je že primerna za realizacijo z dvovhodnimi vpoglednimi tabelami (ang. look-up table) v vezju FPGA in je ni treba posebej predelovati.

$$f(x_1, x_2, x_3, x_4) = (x_1 \equiv x_3) \oplus (x_2 \equiv x_4)$$

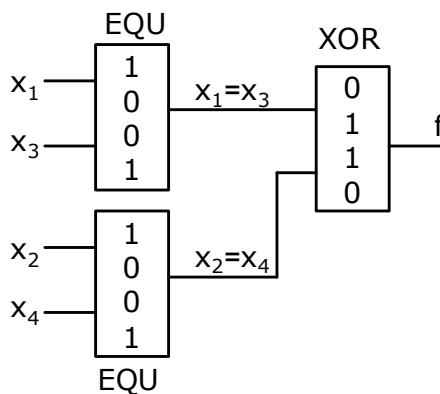
Dvovhodne vpogledne tabele (LUT2) so sestavljene iz pomnilnika (4 RAM celice) in treh 2/1 izbiralnikov. V vsako RAM celico so vpisane vrednosti, ki realizirajo eno od 16 osnovnih dvovhodnih funkcij.



$x_1$	$x_2$	AND	OR	XOR	EQU
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1



Na zgornji sliki sta prikazani struktura dvovhodne vpogledne tabele (levo) in posplošeni simbol, ki ga uporabljamo pri risanju realizacij funkcij (desno) ter primer vsebine RAM celic za nekaj osnovnih funkcij (XOR, EQU). Tako lahko LUT uporabljamo za realizacijo funkcij v več nivojih.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki MDNO.

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji so zato primerne čimbolj nenormalne oblike (večnivojske oblike), samo da vsebujejo operatorje ene vrste. Podana funkcija je v MDNO, zato za neposredno realizacijo s 4-bitno ALU ni primerna, saj vsebuje operaciji AND in OR – torej bi za realizacijo rabili najmanj dve aritmetični–logični enoti in tretjo za izvedbo inverterjev. Funkcijo MDNO prevedemo na operator enega tipa – operator NAND, kar pomeni obliko SNO (Sheffer–jeva normalna oblika funkcije):

$$f(a,b,c,d,e) = a \cdot b \cdot d + \bar{c} \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

Najprej pri prvih dveh členih izpostavimo člen  $d$ , saj petih operacij z eno 4 bitno ALU ne moremo izvesti.

$$f(a,b,c,d,e) = (a \cdot b + \bar{c}) \cdot d + \bar{e}$$

$$f(a,b,c,d,e) = \overline{\overline{(a \cdot b + \bar{c})} \cdot d + \bar{e}}$$

Za pretvorbo v SNO nad vsemi konjunkcijami izvedemo dvojno negacijo. Nad členom v oklepaju uporabimo De Morganov teorem, da dobimo izražavo z NAND operatorjem.

$$f(a,b,c,d,e) = \overline{\overline{\overline{(a \cdot b)} \cdot c} \cdot d + \bar{e}}$$

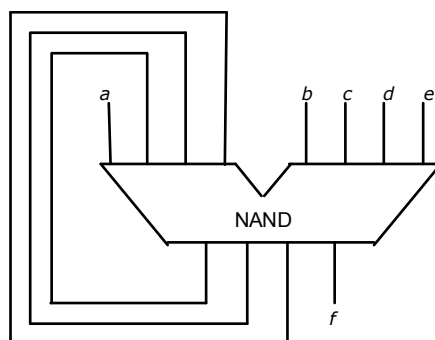
Podobno storimo še enkrat:

$$f(a,b,c,d,e) = \overline{\overline{\overline{\overline{(a \cdot b)} \cdot c} \cdot d} \cdot e}$$

Dobljene NAND operatorje predstavimo z oklepaji.

$$f(a,b,c,d,e) = (((a \uparrow b) \uparrow c) \uparrow d) \uparrow e$$

Narišemo realizacijo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števcov, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF:  
 Iz stolpca T<sub>0</sub> se vidi, da je T<sub>0</sub>='1'. Iz stolpca T<sub>1</sub> se vidi, da se ponavlja

vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T <sub>1</sub>
0	Q <sub>0</sub>
1	Q <sub>0</sub> '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

Za T<sub>2</sub> se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

SMER				
Q <sub>2</sub>	1	0	0	0
	0	0	1	0
	0	0	1	0
	1	0	0	0
	Q <sub>1</sub>			Q <sub>0</sub>

$$T_2 = SMER \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{SMER} \cdot Q_1 \cdot$$

$$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$$

V enačbi za T<sub>2</sub> poiščemo podobnosti z enačbo za T<sub>1</sub>: Enačba za T<sub>1</sub> vsebuje konjunkciji SMER·Q<sub>0</sub>' in SMER'·Q<sub>0</sub>, ki sta vsebovani tudi v enačbi za T<sub>2</sub>, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191<sup>1</sup>. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74191>

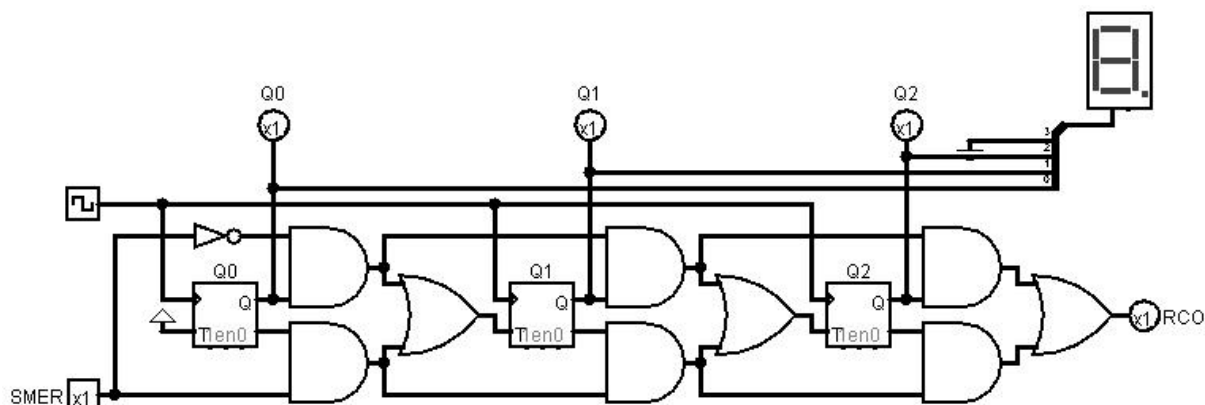
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca (RCO – oz. ripple carry out, včasih tudi TC – terminal count). RCO je signal, ki postane '1' ob prehodu iz najvišjega stanja števca (v našem primeru je to "111") v stanje "000" pri štetju navzgor in ob prehodu "000" v najvišje stanje števca pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števec vežemo kaskadno – torej da signal RCO vežemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu  $Q_2$ , kot kaže spodnja slika:



Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_up\_down\_3\_bit\_using\_T\_FF.circ

Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

- desetiški (BCD) števci, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161<sup>2</sup>.

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74161>

Rešitev 4. naloge:

V prvi iteraciji zberemo skupaj vsa stanja v enem razdelku:  $P_1 = (ABCDEFGG)$

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Naslednja iteracija loči stanja, ki imajo različne izhode:  $P_2 = (ABD)(CEFG)$

- Pregledamo vsa naslednja stanja pri vhodu 0 in 1 v vsakem bloku:
  - Blok (ABD):
    - Naslednja stanja pri w=0 (BDB)
    - Naslednja stanja pri w=1 (CFG)
  - Blok (CEFG):
    - Naslednja stanja pri w=0 (FFEF)
    - Naslednja stanja pri w=1 (ECDG)

Vsa stanja niso v enem bloku. Problem je pri stanju F, ki ima naslednje stanje D. Zato bo stanje F NEEKVIVALENTNO ostalim CEG.

- Novo stanje F zato postavimo v svojo skupino.

Naslednja iteracija loči stanje F od ostalih  $P_3 = (ABD)(CEG)(F)$

- Blok (ABD):
  - Naslednja stanja pri w=0 (BDB)  
So vsa v istem bloku
  - Naslednja stanja pri w=1 (CFG) Niso v istem bloku, ker je F v drugem bloku kot C in G. Zato bo stanje B v novem bloku.
- Blok (CEG):
  - Naslednja stanja pri w=0 (FFF)
  - Naslednja stanja pri w=1 (ECG) C, E in G imamo lahko še vedno za ekvivalentna

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

<i>F</i>	<i>E</i>	<i>D</i>	<i>0</i>
<i>G</i>	<i>F</i>	<i>G</i>	<i>0</i>

Naslednja iteracija loči stanje B od ostalih  $P_4 = (AD)(B)(CEG)(F)$

- Blok (*AD*)
  - Naslednja stanja pri  $w=0$  (*BB*)
  - Naslednja stanja pri  $w=1$  (*CG*)
  - So vsa v istem bloku.
- Blok (*CEG*)
  - Naslednja stanja pri  $w=0$  (*FFF*)
  - Naslednja stanja pri  $w=1$  (*ECG*) So vsa v istem bloku.

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
<i>A</i>	<i>B</i>	<i>C</i>	<i>1</i>
<i>B</i>	<i>D</i>	<i>F</i>	<i>1</i>
<i>C</i>	<i>F</i>	<i>E</i>	<i>0</i>
<i>D</i>	<i>B</i>	<i>G</i>	<i>1</i>
<i>E</i>	<i>F</i>	<i>C</i>	<i>0</i>
<i>F</i>	<i>E</i>	<i>D</i>	<i>0</i>
<i>G</i>	<i>F</i>	<i>G</i>	<i>0</i>

$P_5 = (AD)(B)(CEG)(F)$

Iteraciji  $P_5$  in  $P_4$  sta enaki, zato se postopek minimizacije zaključi. Stanji *A* in *D* sta ekvivalentni. Stanja *C*, *E* in *G* so ekvivalentna.

- Tabelo stanj zapišemo na novo
- Izbrišemo vrstice za *D*, *E* in *G*
- Zamenjamo stanja:  $D \rightarrow A$  in vse  $E \rightarrow C$  ter  $G \rightarrow C$

Rezultat je nova tabela stanj minimiziranega avtomata:

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
<i>A</i>	<i>B</i>	<i>C</i>	<i>1</i>
<i>B</i>	<i>A</i>	<i>F</i>	<i>1</i>
<i>C</i>	<i>F</i>	<i>C</i>	<i>0</i>
<i>F</i>	<i>C</i>	<i>A</i>	<i>0</i>

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

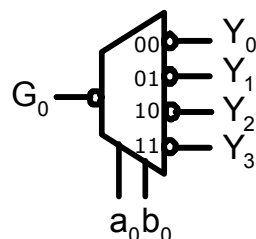
# RAZVOJ DIGITALNIH SISTEMOV

Izpit

13. 02. 2017

1. Realizirajte funkcijo  $f^3 = V(1, 2, 4, 7)$  z redundantnimi mintermi pri  $V_x(0, 3, 6)$  z enim TTL dekoderjem 74139. Dekoder 74139 ima vhod za omogočenje elementa ( $G$ ) in izhode  $Y_0, Y_1, Y_2, Y_3$  v negativni logiki. Njegovo delovanje povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

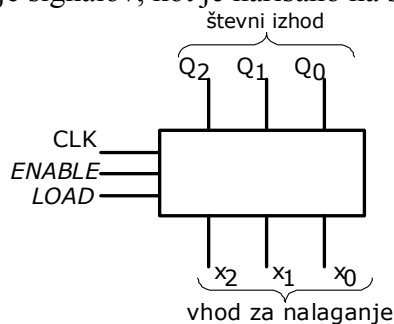


2. Realizirajte podano funkcijo  $f$  z redundancami z eno 4-bitno aritmetično-logično enoto (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Prikažite sintezo 3-bitnega sinhronega števca navzgor z omogočanjem štetja ( $ENABLE$ ) in vzporednim nalaganjem ( $LOAD$ ) z D flip-flopi, izbiralniki 2/1 in logičnimi vrati. Logika vseh krmilnih signalov je pozitivna. S tovrstnim števcem realizirajte števec, ki šteje 2, 3, 4, 5, 2, 3, 4, 5 ...

Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



4. Pretvorite podani avtomat končnih stanj, ki je podan v Mealy-evi obliki, v Moore-ovo obliko.

Trenutno stanje	Naslednje stanje	
	$x1$	$x2$
S1	S2/y1	S3/y0
S2	S5/y1	S3/y1
S3	S1/y0	S5/y0
S4	S2/y1	S4/y1
S5	S3/y0	S2/y1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



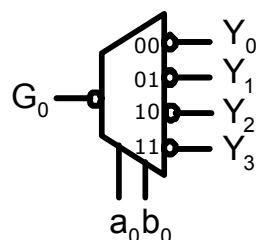
# DEVELOPMENT OF DIGITAL SYSTEMS

Written examination

13. 02. 2017

- Implement a given function  $f^3 = V(1, 2, 4, 7)$  with redundant minterms  $V_x(0, 3, 6)$  using a single TTL decoder 74139. Decoder 74139 features an enable input (G) and outputs  $Y_0, Y_1, Y_2, Y_3$  in negative logic. Its operation and symbol are summarized below:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

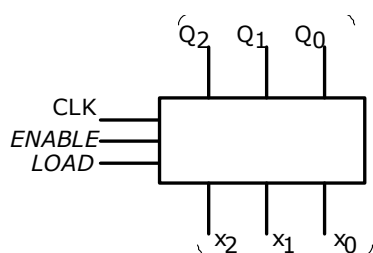


- Implement a given function  $f$  using a single 4-bit arithmetic logic unit (ALU). Any resulting negations of variables must be implemented within this ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

- Implement a synchronous 3 bit *up* binary counter using D type flip-flops, multiplexers 2/1 and logic gates: Design the state transition table, determine the flip-flop input equations and draw the resulting circuit using signal names as specified below. The counter has a count enable input (*ENABLE*), a parallel load signal (*LOAD*), which loads a 3-bit input  $x_2, x_1, x_0$  into 3-bit output  $Q_2, Q_1, Q_0$ . Name the signals according to figure below.

Using the this element, implement a counter, which will count in sequence 2, 3, 4, 5, 2, 3, 4, 5 ...



- Convert a given Mealy type finite state machine into its functionally equivalent Moore form.

Current state	Next state	
	$x1$	$x2$
S1	S2/y1	S3/y0
S2	S5/y1	S3/y1
S3	S1/y0	S5/y0
S4	S2/y1	S4/y1
S5	S3/y0	S2/y1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

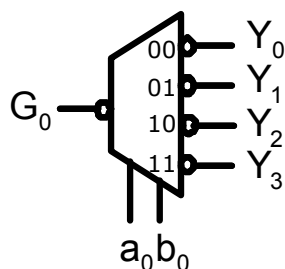
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# Rešitev 1. naloge:

Delovanje dekoderja 74139<sup>1</sup> povzema spodnja tabela:

$G_0$	$a_0$	$b_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Funkcijo  $f$  narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

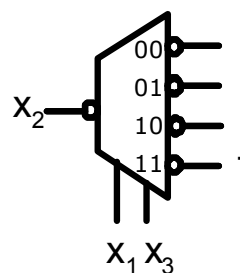
	$x_1$			
$x_2$	X	1	X	1
	1	0	1	X
	$x_3$			

Čim imamo na voljo dekodirnik z ENABLE vhomom v negativni logiki preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '1' vključno z redundancami X.

V zgornjem Veitch–evem diagramu je to spremenljivka  $x_2$ : Namreč, če je  $x_2=1$ , potem lahko vse redundance izberemo tako, da bo  $f=1$  za vse vrednosti  $x_2=1$ . Ko določimo spremenljivko za omogočenje elementa ( $G$ ), opazujemo samo preostali del Veitch–evega diagrama. Spodnje 4 vrednosti diagrama narišemo v novem Veitch–evem diagramu 2 spremenljivk.

	$x_1$	
$x_3$	0	1
	1	X

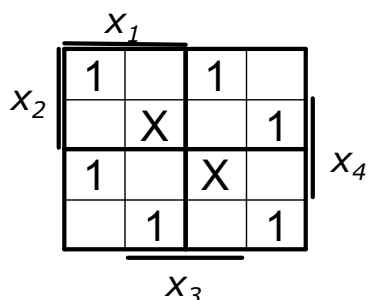
Dekoder ima aktivno nizke izhode, zato iz nastalega diagrama realiziramo *negacijo* funkcije  $f$  zato v Veitch–evem diagramu združujemo ničle, kar nastopa samo v primeru ko sta  $x_1=1$  in  $x_3=1$ .



<sup>1</sup><http://www.alldatasheet.com/view.jsp?Searchword=74HC139>

## Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:



Funkcija vsebuje same diagonalne člene, zato realizacija v obliki KNO oz. DNO ne nudi minimalne oblike. Če se izkaže, da je funkcija linearna, jo lahko realiziramo s pomočjo XOR funkcij. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

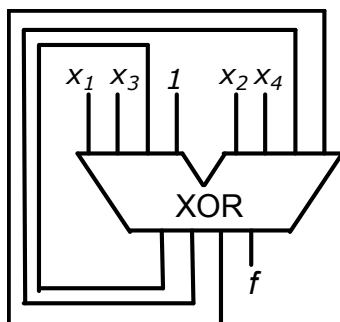
Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

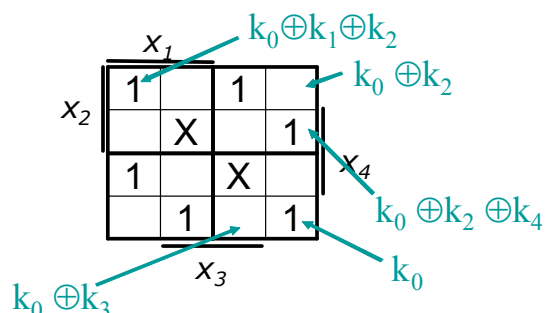
$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji uporabimo lastnost združevanja, ki velja za XOR funkcijo.

$$f(x_1, x_2, x_3, x_4) = 1 \oplus ((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$$



Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

### Rešitev 3. naloge:

trenutno stanje			naslednje stanje			D-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Podobno za D<sub>2</sub> narišemo Veitchev diagram

$D_2$ :

	Q <sub>2</sub>			
Q <sub>1</sub>	1	0	1	0
	1	1	0	0
	Q <sub>0</sub>			

Za D<sub>2</sub> sledi:

$$D_2 = Q_2 Q_1' + Q_2 Q_0' + Q_2' Q_1 Q_0$$

Iz tabele prehajanja stanj števca določimo enačbe D-FF:

Za D<sub>0</sub> se iz tabele vidi  $D_0 = Q_0'$

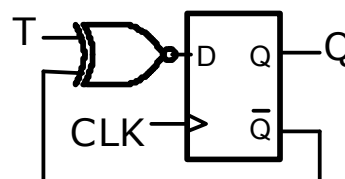
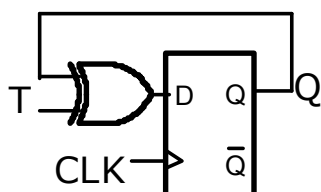
Za D<sub>1</sub> narišemo Veitchev diagram:

$D_1$ :

	Q <sub>2</sub>			
Q <sub>1</sub>	1	0	0	1
	0	1	1	0
	Q <sub>0</sub>			

$$D_1 = Q_0 \oplus Q_1$$

Če enačbo  $D_0 = Q_0'$  zapišemo kot  $D_0 = 1 \oplus Q_0$  in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:



Slika 1: Realizacija T-FF s pomočjo D-FF in XOR vrat (levo) in XNOR vrat (desno)

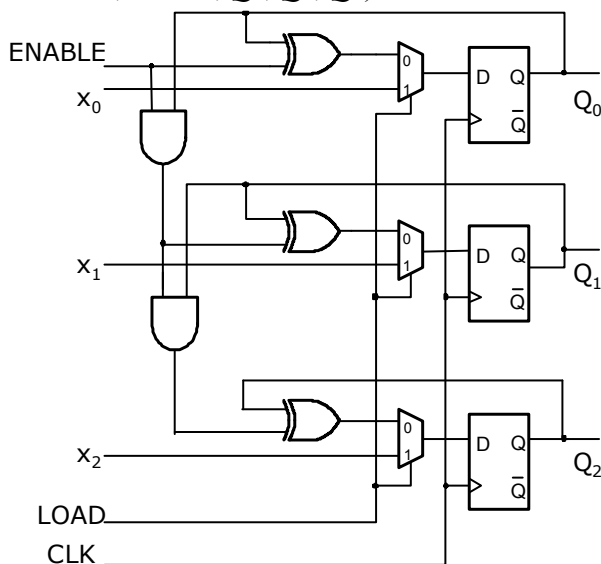
Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (*ENABLE*): Če prvemu "T-FF" (D-FF z XOR vrati) postavimo vhod  $T_0 = '0'$  namesto  $T_0 = '1'$ , vsi FF ne bodo šteli, ampak bodo ohranjali stanje. Torej, če na vhod  $T_0$  postavimo zunanji signal *ENABLE*, števec ne bo štel, ampak ohranjal stanje, če bo *ENABLE* = '0'. V verigi sinhronnega števca so namreč vsi FF vezani tako, da so odvisni od prvega FF. Če stanje ohranja prvi, ga bodo tudi vsi ostali.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnege vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame za 5 spremenljivk ( $ENABLE$ ,  $LOAD$ ,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



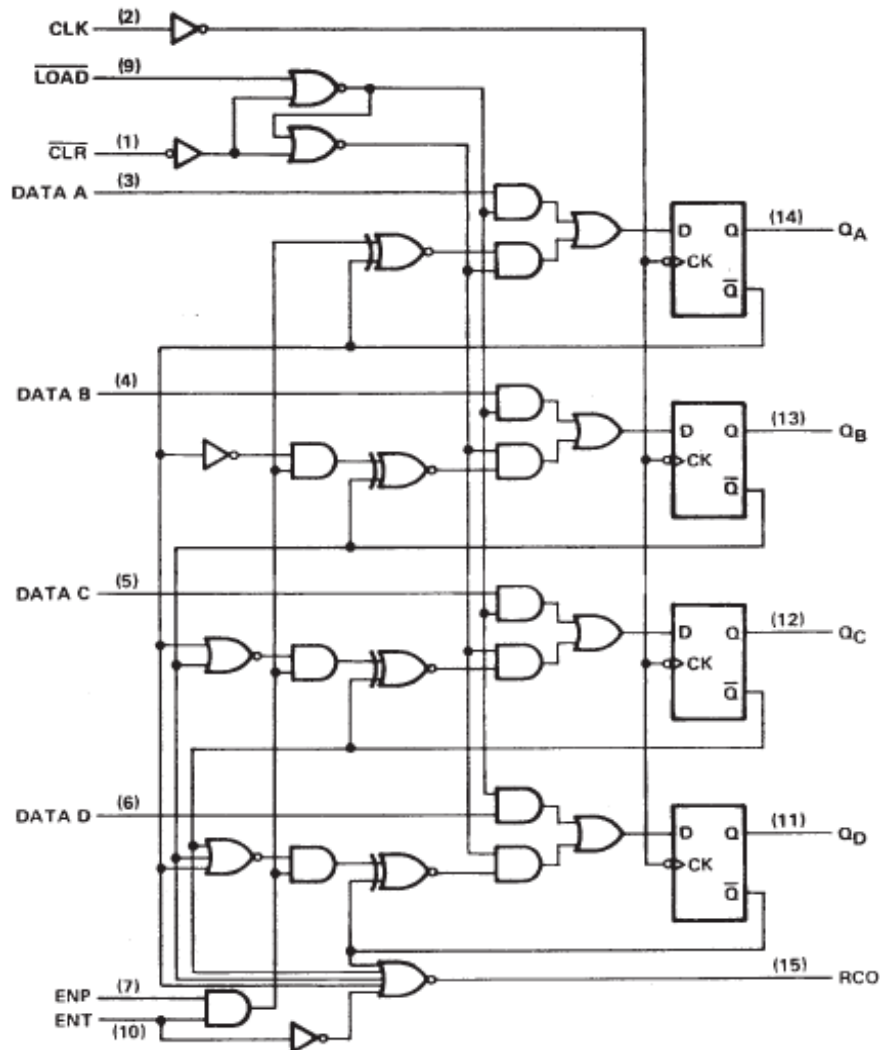
**Slika 2: Sinhroni števec z vzporednim nalaganjem (LOAD) in omogočanjem štetja (ENABLE) (3-bitna izvedba).**

Če želimo z nastalim števcem šteti naraščajoče 2, 3, 4, 5 ..., moramo ko števec pride do stanja 5 ( $Q_2Q_1Q_0=101_2$ ) števec postaviti nazaj na stanje v stanje 2 ( $Q_2Q_1Q_0=x_2x_1x_0=010_2$ ), torej signal  $LOAD$  dekodiramo s pomočjo 2–vhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se  $Q_2 = '1'$  in  $Q_0 = '1'$  v števeni sekvenci pojavlja samo enkrat – če bi se večkrat, bi morali dekodirati tudi  $Q_1$ . Pri tovrstnih števcih ponavadi uporabljamo še en signal  $RESET$ , s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod D–FF za brisanje asinhrono priključimo signal  $RESET$ .

Nastali strukturi števca bi lahko na isti način dodali še četrti bit. Tako bi dobili podobno strukturo kot je 4–bitni sinhroni števec z vzporednim nalaganjem 74163<sup>2</sup>. Pri spodnji realizaciji tega vezja so uporabljeni D–FF, proženi na negativni rob signala ure ( $CLK$ ). Štetje omogočimo s signaloma  $ENP$ ,  $ENT$  (ang. enable parallel, enable transfer). Štetje je izvedeno tako, da se D–FF spremenijo v T–FF, kar dosežemo s pomočjo XNOR vrat, ki imajo en vhod vezan na števeni signal ( $ENT=ENP='1'$ ), drug vhod pa na izhod  $Q'$  FF. AND vrata pred XNOR zagotavljajo prenehanje štetja, če velja  $ENT \cdot ENP \neq '1'$ . Na vhodu D–FF je vezan enostaven izbiralnik iz dveh AND vrat, vezanih na OR vrata. Ta izbiralnik določa, ali bo števec štel, ali bo vzporedno naložil vrednost. Zgornja AND vrata izbiralnika so krmiljena s signalom  $LOAD \cdot CLR'$ , spodnja pa z  $LOAD' \cdot CLR'$ . Čim velja, da je  $CLR' = '1'$  in  $LOAD' = '0'$ , se v D–FF asinhrono naloži vsebina na vseh vseh  $Q_D Q_C Q_B Q_A = DATA_D DATA_C DATA_B DATA_A$ , medtem ko dokler velja  $LOAD' = '1'$  in  $CLR' = '1'$ , bo števec štel navzgor. Če je  $CLR' = '0'$ , je na obeh vseh vseh AND vrat izbiralnika '0' in stanje vseh D–FF se asinhrono postavi na  $Q_D Q_C Q_B Q_A = "0000"$ . Števeni izhodi so  $Q_D Q_C Q_B Q_A$ .

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74163>

Izhod *RCO* (*ripple carry out*) se postavi na '1', ko je števec v stanju "1111" ob tem, da je  $ENT=1$ . Signal *RCO* je izveden z NOR vrati na vhode katerih so priključeni negirani izhodi D-FF, kar je po De Morgan-ovem teoremu ekvivalent AND vratom.



Slika 3: Struktura 4-bitnega MSI sinhronega števca z vzporednim nalaganjem (74163).

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

V zapisu vidimo, da je upoštevanih vseh pet stanj podanega Mealyjevega avtomata končnih stanj, zato zapišemo tabelo prehajanja za Moore-ov avtomat z novimi oznakami stanj in njihovimi izhodi. Prehode stanj prepišemo iz tabele Mealyjevega avtomata z novimi oznakami, kjer se vrstice pri stanjih z več različnimi izhodi ponovijo. Izhodi Mooreovega avtomata ustrezajo izhodom, ki smo jih upoštevali pri določanju stanj Mealyjevega avtomata.

Trenutno stanje	Naslednje stanje	
	$x_1$	$x_2$
S <sub>1</sub>	S <sub>2</sub> /y <sub>1</sub>	S <sub>3</sub> /y <sub>0</sub>
S <sub>2</sub>	S <sub>5</sub> /y <sub>1</sub>	S <sub>3</sub> /y <sub>1</sub>
S <sub>3</sub>	S <sub>1</sub> /y <sub>0</sub>	S <sub>5</sub> /y <sub>0</sub>
S <sub>4</sub>	S <sub>2</sub> /y <sub>1</sub>	S <sub>4</sub> /y <sub>1</sub>
S <sub>5</sub>	S <sub>3</sub> /y <sub>0</sub>	S <sub>2</sub> /y <sub>1</sub>

Na vseh prehodih v naslednje stanje pregledamo če ob tem obstaja samo en izhod, oz. ali jih je morda več. Mealy-eva oblika avtomata namreč lahko spreminja izhode glede na trenutno stanje ( $S_x$ ) in vhod ( $x_1$  oz.  $x_2$ ), zato so izhodi vezani na prehode med stanji, medtem ko je pri Moore-ovi obliki izhod vezan samo na stanje.

Zgornji avtomat ima pet stanj in dva izhoda, zato je v splošnem možnih 10 različnih stanj v pretvorjenega Moore-ovega avtomata.

Če pregledamo prehode, ki vodijo v naslednje stanje S<sub>1</sub>, opazimo, da bo izhod v edinem prehodu enak y<sub>0</sub>. V *funkcijsko* ekvivalentnem Moore-ovem avtomatu ima zato stanje S<sub>1</sub> izhod y<sub>0</sub> - zaradi enostavnosti mu damo ime S<sub>10</sub>. Podobno velja za stanje S<sub>2</sub>, ki ga preimenujemo v S<sub>21</sub> in stanje S<sub>4</sub>, ki ga preimenujemo v S<sub>41</sub>.

Stanje S<sub>3</sub> pa ima na prehodih dva možna izhoda: y<sub>0</sub> in y<sub>1</sub>, zato vpeljemo dve novi Moore-ovi stanji S<sub>30</sub>, z izhodom y<sub>0</sub> in S<sub>31</sub>, z izhodom y<sub>1</sub>. Obnašanje avtomata v obeh Moore-ovih stanjih mora ostati enako originalu, zato polji naslednjih stanj v vrstici S<sub>3</sub> prepišemo, le izhod Moore-ovega stanja se ustrezno spremeni (y<sub>0</sub> v primeru S<sub>30</sub> in y<sub>1</sub> v primeru S<sub>31</sub>). Podobno storimo za stanje S<sub>5</sub> in dobimo pretvorjeno, *funkcijsko* ekvivalentno obliko.

Trenutno stanje	Naslednje stanje		Izhod
	$x_1$	$x_2$	
S10	S21	S30	y <sub>0</sub>
S21	S51	S31	y <sub>1</sub>
S30	S10	S50	y <sub>0</sub>
S31	S10	S50	y <sub>1</sub>
S41	S21	S41	y <sub>1</sub>
S50	S30	S21	y <sub>0</sub>
S51	S30	S21	y <sub>1</sub>

Dobljeni avtomat zgolj *funkcijsko* pooseblja Mealy-evo izvedbo - popolnoma ga ne more nikdar nadomestiti, saj Mealy-vi avtomati na izhodu izkazujejo asinhrono obnašanje (vhod - in posledično tudi izhod - se namreč lahko spremeni neodvisno od ure). Tega Moore-ov avtomat ne izkazuje, saj v naslednje stanje preide le ob aktivnem robu signala ure, zato se Moore-ov *funkcijski* ekvivalent vedno obnaša počasneje - za en cikel signala ure. Slednji je namreč potreben, da pretvorjeni avtomat preide v novo Moore-ovo stanje (npr. S31) in spremeni izhod (npr. na y1).

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

25. 08. 2017

1. Realizirajte funkcijo  $f$  s čim manj izbiralniki 4/1.

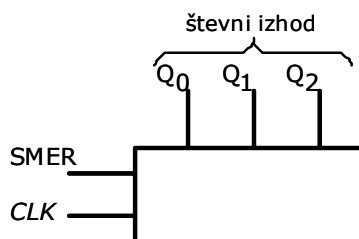
$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2 = x_1 \cdot x_2 \cdot x_3$
- $f_3$  = funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh dveh.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko. Vezje PAL je narisano na hrbtni strani izpita.

3. Prikažite sintezo sinhronnega dvosmernega 3-bitnega števca z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov. Števec ima vhod SMER, ki določa smer štetja: Če je SMER='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Minimizirajte podani avtomat končnih stanj z uporabo metode z razdelki ter zapišite tabelo prehajanja stanj nastalega minimalnega avtomata.

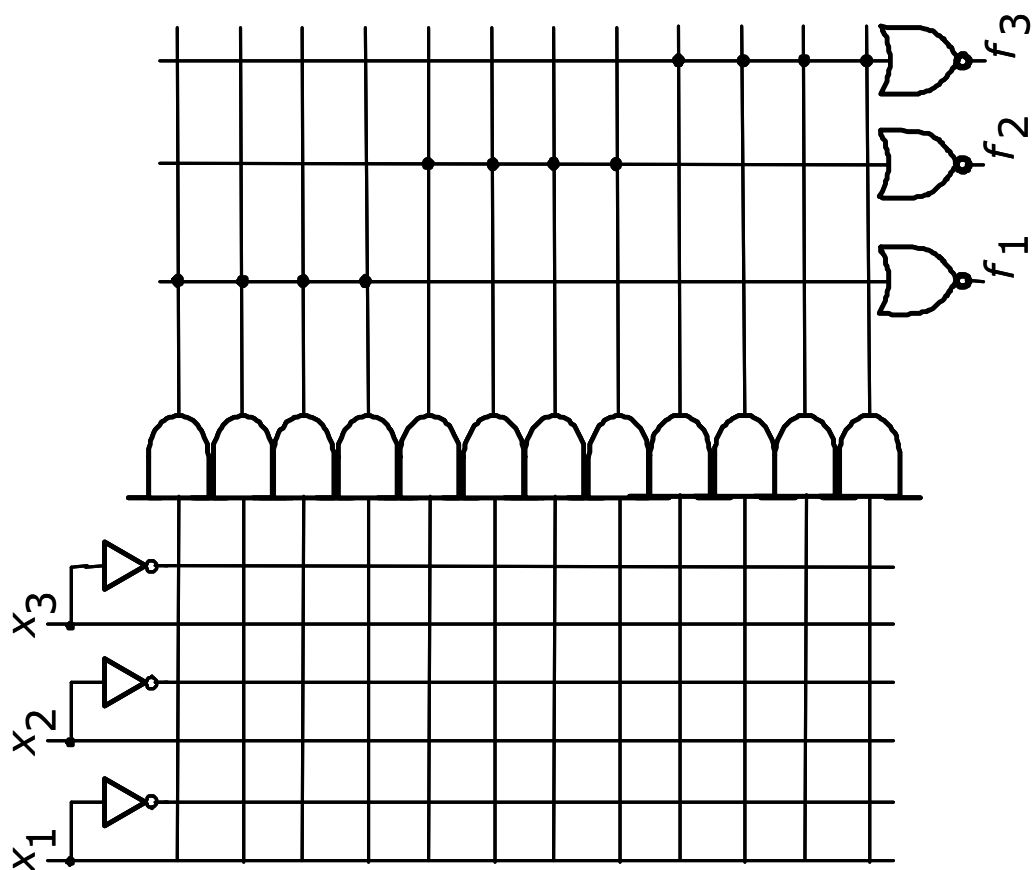
<i>Trenutno stanje</i>	<i>Naslednje stanje</i>		<i>Izhod</i>
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

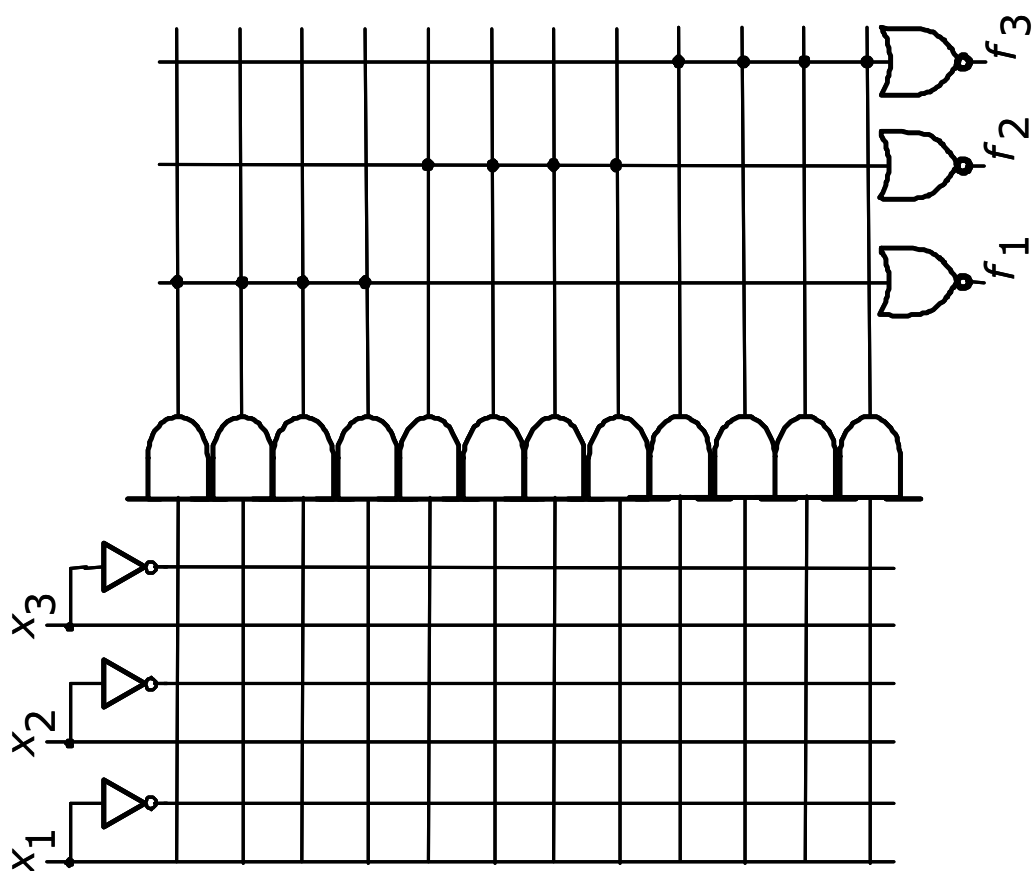
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.





Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

Rešitev 1. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

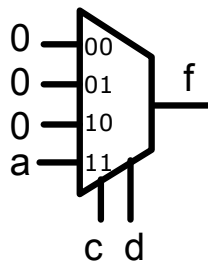
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

$$f_{PDNO}(a,b,c,d) = V(11,15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

## Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\begin{aligned} f_2 &= \overline{x_1 \cdot x_2 \cdot x_3} \\ \overline{f_2} &= \overline{x_1} + \overline{x_2} + \overline{x_3} \end{aligned}$$

Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

$$\overline{f_3} = \overline{x_1} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vezemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6–vhodna. Vezje PAL3L3 je AND–NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števcov, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF: Iz stolpca T<sub>0</sub> se vidi, da je T<sub>0</sub>='1'. Iz stolpca T<sub>1</sub> se vidi, da se ponavlja

vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T <sub>1</sub>
0	Q <sub>0</sub>
1	Q <sub>0</sub> '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

Za T<sub>2</sub> se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

SMER					
Q <sub>2</sub>	1	0	0	0	Q <sub>0</sub>
	0	0	1	0	
	0	0	1	0	
	1	0	0	0	
					Q <sub>1</sub>

$$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$$

V enačbi za T<sub>2</sub> poiščemo podobnosti z enačbo za T<sub>1</sub>: Enačba za T<sub>1</sub> vsebuje konjunkciji SMER·Q<sub>0</sub>' in SMER'·Q<sub>0</sub>, ki sta vsebovani tudi v enačbi za T<sub>2</sub>, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191<sup>1</sup>. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74191>

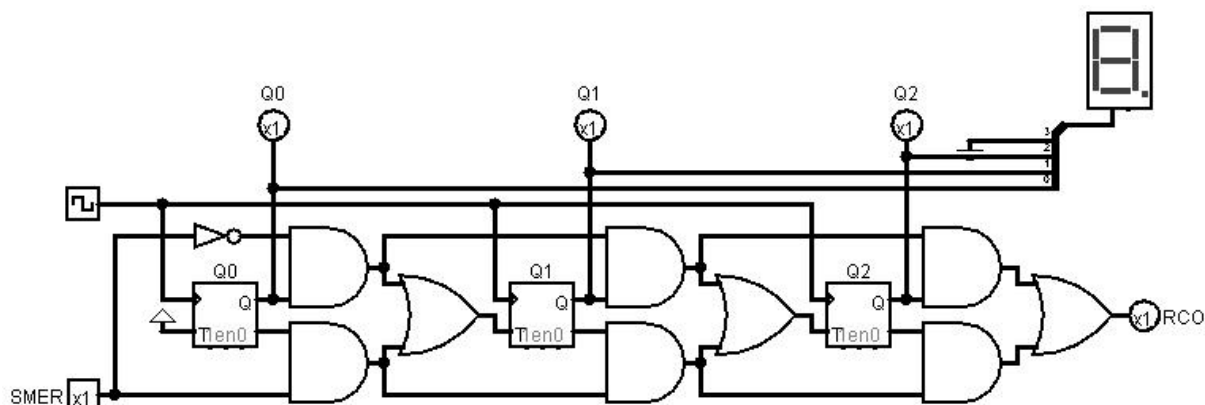
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca (RCO – oz. ripple carry out, včasih tudi TC – terminal count). RCO je signal, ki postane '1' ob prehodu iz najvišjega stanja števca (v našem primeru je to "111") v stanje "000" pri štetju navzgor in ob prehodu "000" v najvišje stanje števca pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števce vezemo kaskadno – torej da signal RCO vezemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu  $Q_2$ , kot kaže spodnja slika:



Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_up\_down\_3\_bit\_using\_T\_FF.circ

Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

- desetiški (BCD) števci, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161<sup>2</sup>.

<sup>2</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74161>

Rešitev 4. naloge:

V prvi iteraciji zberemo skupaj vsa stanja v enem razdelku:  $P_1 = (ABCDEFGG)$

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Naslednja iteracija loči stanja, ki imajo različne izhode:  $P_2 = (ABD)(CEFG)$

- Pregledamo vsa naslednja stanja pri vhodu 0 in 1 v vsakem bloku:
    - Blok (ABD):
      - Naslednja stanja pri w=0 (BDB)
      - Naslednja stanja pri w=1 (CFG)
    - Blok (CEFG):
      - Naslednja stanja pri w=0 (FFEF)
      - Naslednja stanja pri w=1 (ECDG)
- Vsa stanja niso v enem bloku. Problem je pri stanju F, ki ima naslednje stanje D. Zato bo stanje F NEEKVIVALENTNO ostalim CEG.
- Novo stanje F zato postavimo v svojo skupino.

Naslednja iteracija loči stanje F od ostalih  $P_3 = (ABD)(CEG)(F)$

- Blok (ABD):
  - Naslednja stanja pri w=0 (BDB)  
So vsa v istem bloku
  - Naslednja stanja pri w=1 (CFG) Niso v istem bloku, ker je F v drugem bloku kot C in G. Zato bo stanje B v novem bloku.
- Blok (CEG):
  - Naslednja stanja pri w=0 (FFF)
  - Naslednja stanja pri w=1 (ECG) C, E in G imamo lahko še vedno za ekvivalentna

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni v sistemu STUDIS.

<i>F</i>	<i>E</i>	<i>D</i>	<i>0</i>
<i>G</i>	<i>F</i>	<i>G</i>	<i>0</i>

Naslednja iteracija loči stanje B od ostalih  $P_4=(AD)(B)(CEG)(F)$

- Blok (*AD*)
  - Naslednja stanja pri  $w=0$  (*BB*)
  - Naslednja stanja pri  $w=1$  (*CG*)
  - So vsa v istem bloku.
- Blok (*CEG*)
  - Naslednja stanja pri  $w=0$  (*FFF*)
  - Naslednja stanja pri  $w=1$  (*ECG*) So vsa v istem bloku.

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
<i>A</i>	<i>B</i>	<i>C</i>	<i>1</i>
<i>B</i>	<i>D</i>	<i>F</i>	<i>1</i>
<i>C</i>	<i>F</i>	<i>E</i>	<i>0</i>
<i>D</i>	<i>B</i>	<i>G</i>	<i>1</i>
<i>E</i>	<i>F</i>	<i>C</i>	<i>0</i>
<i>F</i>	<i>E</i>	<i>D</i>	<i>0</i>
<i>G</i>	<i>F</i>	<i>G</i>	<i>0</i>

$P_5=(AD)(B)(CEG)(F)$

Iteraciji  $P_5$  in  $P_4$  sta enaki, zato se postopek minimizacije zaključi. Stanji *A* in *D* sta ekvivalentni. Stanja *C*, *E* in *G* so ekvivalentna.

- Tabelo stanj zapišemo na novo
- Izbrišemo vrstice za *D*, *E* in *G*
- Zamenjamo stanja:  $D \rightarrow A$  in vse  $E \rightarrow C$  ter  $G \rightarrow C$

Rezultat je nova tabela stanj minimiziranega avtomata:

Trenutno stanje	Naslednje stanje		Izhod z
	w=0	w=1	
<i>A</i>	<i>B</i>	<i>C</i>	<i>1</i>
<i>B</i>	<i>A</i>	<i>F</i>	<i>1</i>
<i>C</i>	<i>F</i>	<i>C</i>	<i>0</i>
<i>F</i>	<i>C</i>	<i>A</i>	<i>0</i>

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni v sistemu STUDIS.

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij 08. 12. 2010

1. Določite minimalno normalno obliko (MNO) funkcije  $f$ .

$$f^A = V(0, 1, 5, 6, 7, 13, 14, 15)$$

2. Ali je funkcija  $f$  linearna? Če je linearna, potem izračunajte koeficiente linearnosti. Če ni linearna, potem utemeljite zakaj.

$$f^A = V(1, 3, 4, 6, 9, 11, 12, 14)$$

3. Prikažite postopek deljenja nepredznačenih števil v dvojiškem sistemu na primeru:

$$38_{10} / 9_{10}$$

4. Pretvorite število  $4B_{16}$  v BCD zapis z uporabo "double dabble" algoritma.

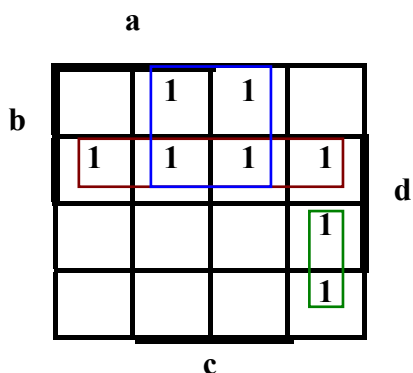


## Rešitev 1. naloge

Funkcija je zapisana v PDNO. Potrebno je določiti MNO. Za to najprej določimo MDNO in MKNO obliki, nato preštajemo število operatorjev in vrat (COST funkcije) in tisto realizacijo, ki ima manj operatorjev (MDNO, MKNO) izberemo kot MNO. Če sta COST funkciji obeh oblik enakovredni, je MNO lahko MDNO ali MKNO.

PDNO:  $f^4 = V(0, 1, 5, 6, 7, 13, 14, 15)$

**MDNO:**

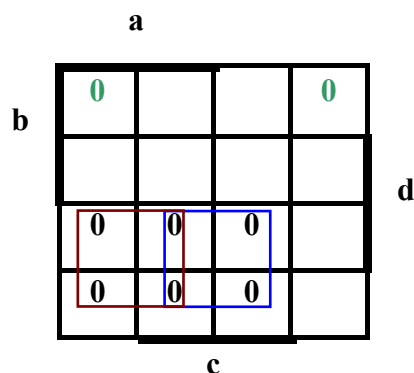


$$f_{MDNO}^4 = b \cdot c + b \cdot d + a' \cdot b' \cdot c'$$

$COST = (4 \text{ vrata}, 10 \text{ vhodov})$

brez inverterjev

**MKNO:**



Funkcija MKNO je **negirana**, ker smo zbirali ničle:

$$f_{MKNO}^4 = (a \cdot b' + b' \cdot c + b \cdot c' \cdot d)'$$

zato uporabimo še De Morganov teorem in dobimo

$$f_{MKNO}^4 = (a' + b) (b + c') (b' + c + d)$$

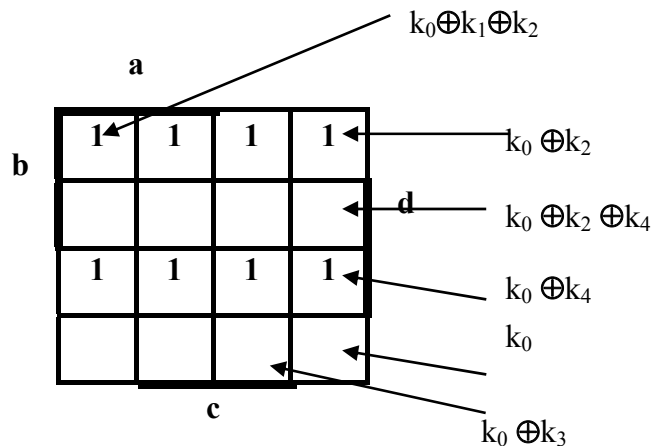
$COST = (4 \text{ vrata}, 10 \text{ vhodov})$

brez inverterjev

Obe obliki, MDNO in MKNO sta enakovredni, saj sta COST funkciji enaki.

## Rešitev 2. naloge:

Potrebno je bilo določiti koeficiente linearnosti funkcije podane v PDNO. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat (torej da **d** postane 1 v prvi iteraciji). Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Prepogibanje je prikazano v knjigi, stran 79, vaja 6.5.5.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(a, b, c, d) = k_0 \oplus k_1 \cdot a \oplus k_2 \cdot b \oplus k_3 \cdot c \oplus k_4 \cdot d \quad (2.1)$$

S pomočjo Veitchevega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=0$  in  $k_0 \oplus k_4=1$ , kar pomeni  $0 \oplus k_4=1 \rightarrow k_4=1$ .

$k_0 \oplus k_2 \oplus k_4=0$ , kar pomeni  $0 \oplus k_2 \oplus 1=0 \rightarrow k_2=1$ .

Če analiziramo naprej dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $0 \oplus k_1 \oplus 1=1 \rightarrow k_1=0$ .

In če napišemo še eno enačbo za  $k_0 \oplus k_3=0$ , kar pomeni  $0 \oplus k_3=0$  sledi da je  $k_3=0$ .

Rešitev enostavno lahko preverimo, če izpišemo MDNO funkcije:

$$f_{MDNO} = b \cdot d' + b' \cdot d = b \oplus d$$

Iz česar lahko ob primerjavi z enačbo (2.1) takoj preberemo taiste koeficiente (samo tista koeficienta pri **b** in **d** sta enaka 1, ostali so 0):

$$k_0=0$$

$$k_1=0$$

$$k_2=1$$

$$k_3=0$$

$$k_4=1$$

Rešitev 3. naloge:

Pretvorimo število v dvojiški zapis  $38_{10}=26_{16}=0010\ 0110_2$

Podobno  $9_{10}=9_{16}=1001_2$

				0	0	1	0	0	kvocient= $4_{10}$				
1	0	0	1	0	0	1	1	0					START
				1	0	0	1						$2 < 9$
				0	0	1	0	0					
					1	0	0	1					$4 < 9$
					0	1	0	0	1				
						1	0	0	1				$9 = 9 \rightarrow$ odštejemo
						0	0	0	0	1			$1 < 9$
							1	0	0	1			
							0	0	0	1	0		$2 < 9$
								1	0	0	1		
								0	0	1	0		ostanek= $2_{10}$

Rešitev 4. naloge:  $4B_{16} = 75_{10}$ . Oziroma zapis posameznih števk: 0111 0101<sub>BCD</sub>.

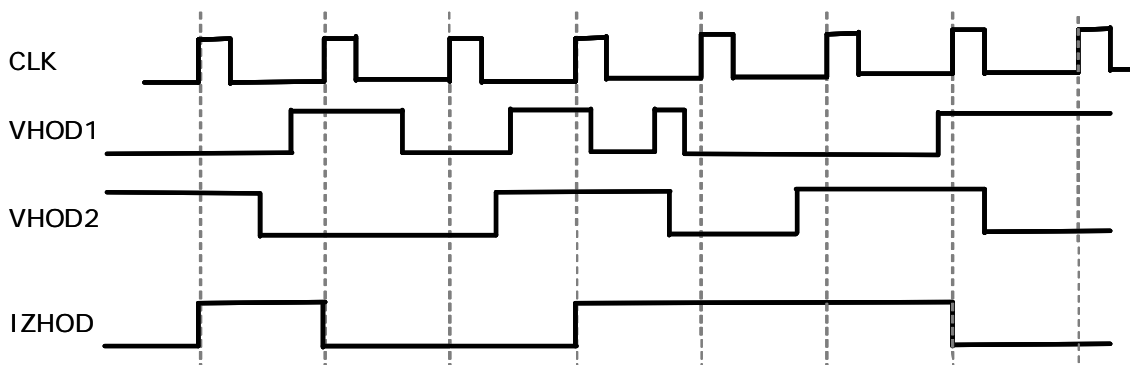
DESETICE				ENICE				0	1	0	0	1	0	1	1	pomik 1
							0	1	0	0	1	0	1	1		pomik 2
						0	1	0	0	1	0	1	1			pomik 3
					0	1	0	0	0	1	0	1	1			pomik 4
				0	1	0	0	1	0	1	1					pomik 5
			0	1	0	0	1	0	1	1						+3
			0	1	1	0	0	0	1	1						pomik 6
		0	1	1	0	0	0	1	1							+3
		0	1	1	0	1	1	1	1							pomik 7
	0	1	1	0	1	1	1	1								+3
	0	1	1	1	0	1	0	1								pomik 8
0	1	1	1	0	1	0	1									
$7_{10}$				$5_{10}$												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

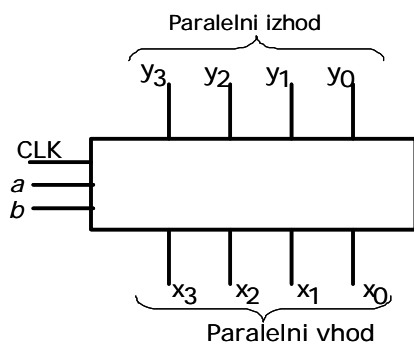
# RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij 19. 1. 2011

1. Prikazano je časovno zaporedje signalov, ki popolnoma podaja delovanje nekega sekvenčnega vezja. Narišite sekvenčno vezje, ki realizira spodnje zaporedje signalov.

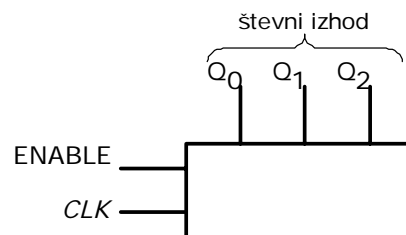


2. S pomočjo 4/1 izbiralnikov, inverterjev in D flip-flopov realizirajte univerzalni register, ki ima dva funkcijska vhoda  $a$  in  $b$  in opravlja funkcije po spodnji tabeli:



$a$	$b$	funkcija
0	0	briše vsebino registra (CLEAR)
0	1	rotira vsebino eno mesto desno (ROR)
1	0	negira vsebino (CPL)
1	1	vpiše vsebino s paralelnega vhoda (LOAD)

3. Prikažite sintezo 3-bitnega sinhronega števca navzgor z omogočanjem štetja (ENABLE) s T flip-flopi in logičnimi vrati. Če je ENABLE='1', števec šteje, če je ENABLE='0' števec ohranja stanje.



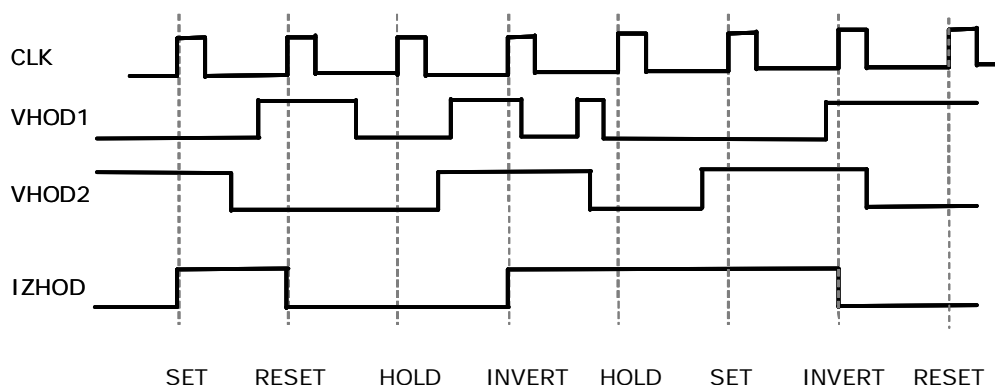
4. Narišite diagram prehajanja stanj avtomata končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z=1$ , ko se na vhodu pojavi zaporedje **101**, sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda.

CLK	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$	$t_{13}$	$t_{14}$	$t_{15}$
$w$	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1	1
$z$	-	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

## Rešitev 1. naloge

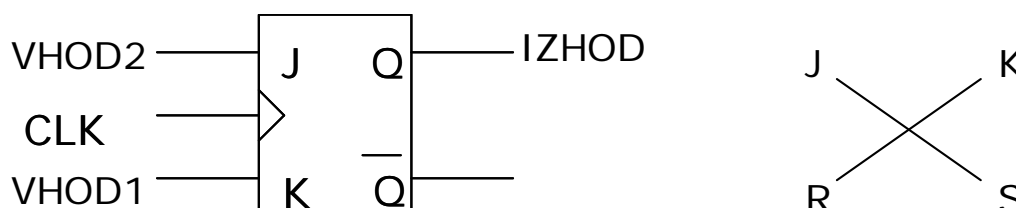
Podano je časovno zaporedje signalov. S črtkano črto je označen rob spremembe, saj gre očitno za robno proženo vezje (pozitivni rob).. Vezje je sekvenčno, torej je delovanje lahko odvisno od vhodov in prejšnjega stanja vezja. Časovni diagram mora vsebovati toliko ciklov signala ure, da je delovanje *popolnoma podano*. V našem primeru je to 8 ciklov, torej lahko sklepamo, da vezje nima več kot eno spominsko stanje, saj dobimo 8 kombinacij s 3 vhodnimi spremenljivkami.



Za vezje narišemo pravilnostno tabelo, v kateri združimo oba vhoda in prejšnje stanje, saj naloga pravi, da gre za sekvenčno vezje.

<i>VHOD1</i>	<i>VHOD2</i>	<i>IZHOD(t)</i>	<i>IZHOD(t+1)</i>	<i>funkcija vezja</i>	<i>JK-FF</i>	
0	0	0	0	HOLD	J=0	K=0
0	0	1	1	HOLD	J=0	K=0
0	1	0	1	SET	J=1	K=0
0	1	1	1	SET	J=1	K=0
1	0	0	0	RESET	J=0	K=1
1	0	1	0	RESET	J=0	K=1
1	1	0	1	INVERT	J=1	K=1
1	1	1	0	INVERT	J=1	K=1

Iz tabele lahko povzamemo, da gre za JK flip-flop, prožen na pozitivni rob signala. VHOD2 ima funkcijo J vhoda (opravlja funkcijo postavljanja – set), medtem ko ima VHOD1 funkcijo K vhoda (opravlja funkcijo brisanja – reset). IZHOD je Q.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

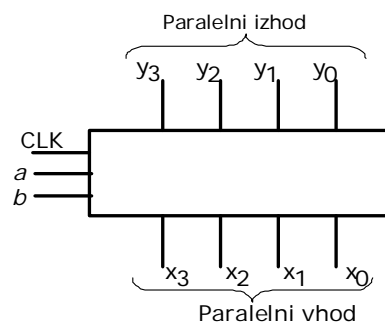
Rešitev 2. naloge:

Naloga zahteva realizacijo univerzalnega registra s funkcijami

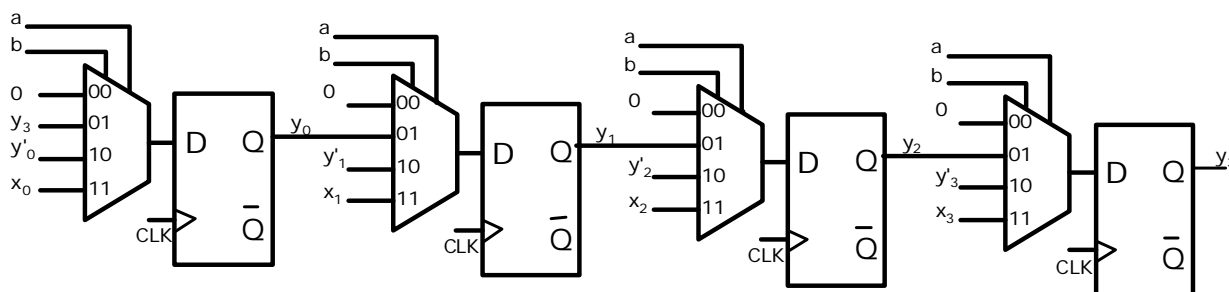
$a$	$b$	<i>funkcija</i>
0	0	briše vsebino registra (CLEAR)
0	1	rotira vsebino eno mesto desno (ROR)
1	0	negira vsebino (CPL)
1	1	vpiše vsebino s paralelnega vhoda (LOAD)

Vsako od operacij izpišemo v pravilnostno tabelo v kateri združimo funkcijska bita  $a$ ,  $b$  in trenutno stanje na  $i$ -tem mestu registra  $y_i(t)$ . Realizacija z D flip-flopi nam analizo močno poenostavi, zaradi enačbe D flip-flopa:  $D = y_{i+1}(t)$

$a$	$b$	$y_i(t)$	$y_i(t+1)$	$D$
0	0	0	0	0
0	0	1	0	0
0	1	0	$y_{i-1}(t)$	$y_{i-1}(t)$
0	1	1	$y_{i-1}(t)$	$y_{i-1}(t)$
1	0	0	$y_i'$	$y_i'$
1	0	1	$y_i'$	$y_i'$
1	1	0	$x_i$	$x_i$
1	1	1	$x_i$	$x_i$



Register izvaja rotacijo, torej nima serijskega vhoda in izhoda, ampak LSB bit  $y_3$  vodimo na MSB bit  $y_0$ . Naloga zahteva realizacijo z 4/1 izbiralniki, s katerimi ločimo 4 operacije registra.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

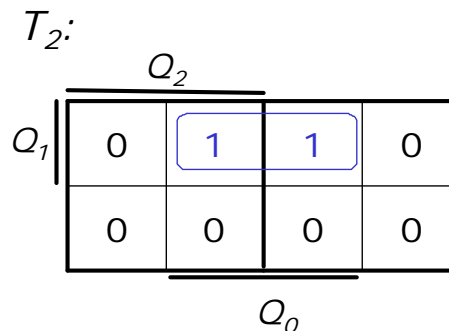
trenutno stanje			naslednje stanje			T-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za T<sub>0</sub> se iz tabele vidi T<sub>0</sub> = 1

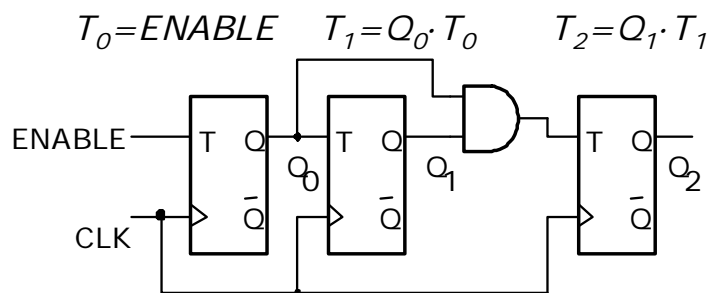
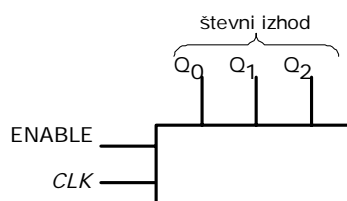
Za T<sub>1</sub> se iz tabele vidi T<sub>1</sub> = Q<sub>0</sub>·1 = Q<sub>0</sub>·T<sub>0</sub>

Za T<sub>2</sub> narišemo Veitchev diagram



$$T_2 = Q_0 \cdot Q_1 = T_1 \cdot Q_1$$

Naloga pravi, da moramo dodati še vhod za omogočanje štetja (ENABLE). Če prvemu T-FF postavimo T<sub>0</sub> = '0' namesto T<sub>0</sub> = '1', flip-flopi števca ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod T<sub>0</sub> postavimo zunanji signal ENABLE, števec ne bo štel, ampak ohranjal stanje, če bo ENABLE='0'. V verigi sinhronnega števca so namreč vezani vsi T-FF tako, da so odvisni od prvega T-FF. Do istega sklepa bi prišli, če bi risali Veitch-eve diagrame za 4 spremenljivke (ENABLE, Q<sub>2</sub>, Q<sub>1</sub>, Q<sub>0</sub>).



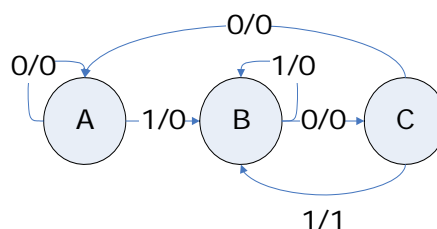
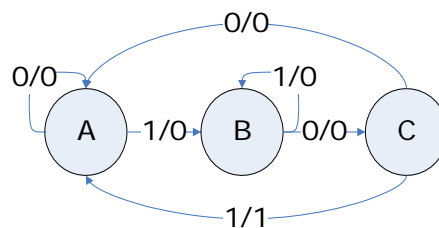
Rešitev 4. naloge:

Iz tabele, ki ponazarja delovanje avtomata končnih stanj se vidi, da to ne more biti Moore–ov tip avtomata ampak Mealy–ev tip, saj se izhod spremeni *takoj*, ko na vhod pride druga '1' sekvence "101", ki jo zaznavamo.

CLK	t <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>	t <sub>10</sub>	t <sub>11</sub>	t <sub>12</sub>	t <sub>13</sub>	t <sub>14</sub>	t <sub>15</sub>
w	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1	1
z	-	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0

Avtomat začnemo risati v nekem začetnem stanju. Za vsako stanje imamo dve možnosti, saj je vhod samo eden. Vezje ostaja v stanju A toliko časa, dokler ne pride na vhod prva '1' zaporedja. Ko se to zgodi, preide avtomat v stanje B, kjer ostane dokler je na vhodu '1' (glej tabelo – trenutek t<sub>9</sub>, t<sub>10</sub>, t<sub>11</sub>). Ko se na vhodu pojavi '0' preide v zadnje stanje, od koder sta dve možnosti: Če se naslednji cikel pojavi '0' moramo do sedaj zaznano zaporedje "10X" zavreči (glej tabelo – trenutek t<sub>2</sub>, t<sub>3</sub>, t<sub>4</sub>), tako da se vrnemo v stanje A, če pa se na vhodu pojavi '1', pa postavimo izhod na '1' in se vrnemo na začetek:

Takšna realizacija ne dovoljuje prekrivanja vzorcev – če dovolimo tudi prekrivanje vzorcev (glej tabelo – trenutek t<sub>7</sub>, t<sub>8</sub>, t<sub>9</sub>), potem mora avtomat preiti iz stanja C ob pogoju w='1' v stanje B in ne v stanje A. Na spodnji sliki sta prikazani obe rešitvi za detektor zaporedja "101": Zgornja slika predstavlja rešitev brez prekrivanja zaporedij, spodnja s prekrivanjem zaporedij.





# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
06. 12. 2011

1. Določite popolno konjunktivno normalno obliko (PKNO) in popolno disjunktivno normalno obliko (PDNO) funkcije  $f$ .

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

2. Ali je funkcija  $f$  linearna? Če je linearna, potem izračunajte koeficiente linearnosti. Če ni linearna, potem utemeljite zakaj.

$$f^4 = V(0, 3, 4, 7, 9, 10, 13, 14)$$

3. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi s čim manj izbiralniki 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 5, 7 - 9, 11, 12) \quad in \quad \&_x(3, 4, 10, 15)$$

4. Pretvorite število  $123_{10}$  ( $1111011_2$ ) v BCD zapis z uporabo "double dabble" algoritma.

## Rešitev 1. naloge

Funkcija je zapisana v večnivojski obliki, torej jo izrazimo v normalno obliko.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

Funkciji NOR ( $\downarrow$ ) in ekvivalence ( $\equiv$ ) izpišemo:

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 + x_2}) \cdot \overline{x_3} + \left( \overline{(\overline{x_2 \oplus x_4}) + x_1} \right)$$

Ekvivalenco smo izrazili kot negacijo XOR. Uporabimo De Morganov teorem:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2 \oplus x_4}) \cdot x_1$$

Izpišemo enačbo funkcije XOR ( $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$ ) in dobimo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2} \cdot \overline{x_4} + x_2 \cdot x_4) \cdot x_1$$

Razširimo še zadnjo konjunkcijo in rezultat je oblika MDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_4} \cdot x_1 + x_1 \cdot x_2 \cdot x_4$$

Če uporabimo lastnost Boole-ove algebre ( $\overline{\overline{a}} + a = 1$ ) lahko zapišemo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4$$

Kar lahko zapišemo v obliki PDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4$$
$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$

PDNO pretvorimo v PKNO tako, da pregledamo manjkajoče minterme: 2, 3, 4, 5, 6, 7, 9, 11, 12, 14. Te minterme preslikamo preko tabele:

m <sub>i</sub>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M <sub>i</sub>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Funkcija v PKNO se torej glasi:

$$f_{PDNO}(x_1, x_2, x_3, x_4) = V(0, 1, 8, 10, 13, 15)$$
$$f_{PKNO}(x_1, x_2, x_3, x_4) = \&(13, 12, 11, 10, 9, 8, 6, 4, 3, 1)$$

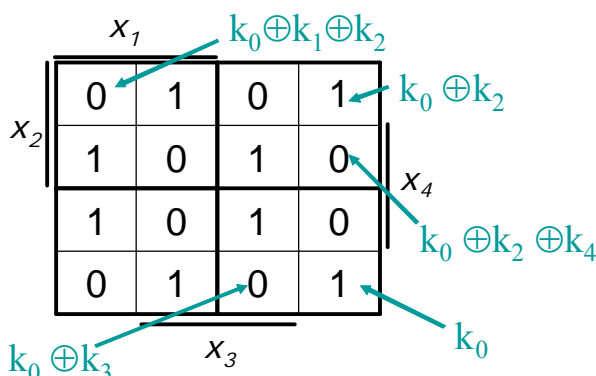
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 2. naloge:

Potrebno je določiti koeficiente linearnosti funkcije podane v PDNO. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat (recimo da  $x_4$  postane 1 v prvi iteraciji). Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Prepogibanje je prikazano v knjigi, stran 79, vaja 6.5.5.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4 \quad (2.1)$$

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $0 \oplus k_3=1 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=1$ , kar pomeni  $1 \oplus k_2=1$  sledi da je  $k_2=0$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=0$ , kar pomeni  $1 \oplus 0 \oplus k_4=0 \rightarrow k_4=1$ .

Če analiziramo naprej dobimo  $k_0 \oplus k_1 \oplus k_2=0$ , kar pomeni  $1 \oplus k_1 \oplus 0=0 \rightarrow k_1=1$ .

Če vstavimo dobljeno v enačbo (2.1) dobimo:  $k_0=1 \quad k_1=1 \quad k_2=0 \quad k_3=1 \quad k_4=1$

In rešitev:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= 1 \oplus x_1 \oplus x_3 \oplus x_4 \\ f(x_1, x_2, x_3, x_4) &= \overline{x_1 \oplus x_3 \oplus x_4} \\ f(x_1, x_2, x_3, x_4) &= (x_1 \equiv x_3 \equiv x_4) \end{aligned}$$

### Rešitev 3. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 5, 7 - 9, 11, 12) \text{ in } \&_x(3, 4, 10, 15)$$

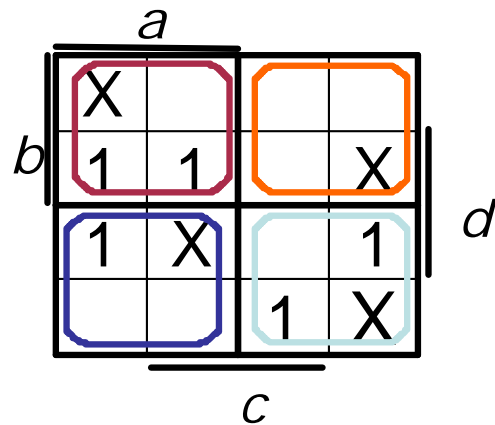
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$a$	$b$	$c$	$d$	$f$
0	15	0	0	0	0	X
1	14	0	0	0	1	1
2	13	0	0	1	0	1
3	12	0	0	1	1	0
4	11	0	1	0	0	0
5	10	0	1	0	1	X
6	9	0	1	1	0	0
7	8	0	1	1	1	0
8	7	1	0	0	0	0
9	6	1	0	0	1	1
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	X
13	2	1	1	0	1	1
14	1	1	1	1	0	0
15	0	1	1	1	1	1

Dobimo:

$$f = V(1, 2, 9, 13, 15) \text{ in } V_x(0, 5, 11, 12)$$

Dobljeno funkcijo vrišemo v Veitch-ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev-em diagramu. Če izberemo kot naslovni spremenljivki  $a$  in  $b$ , potem dobimo:



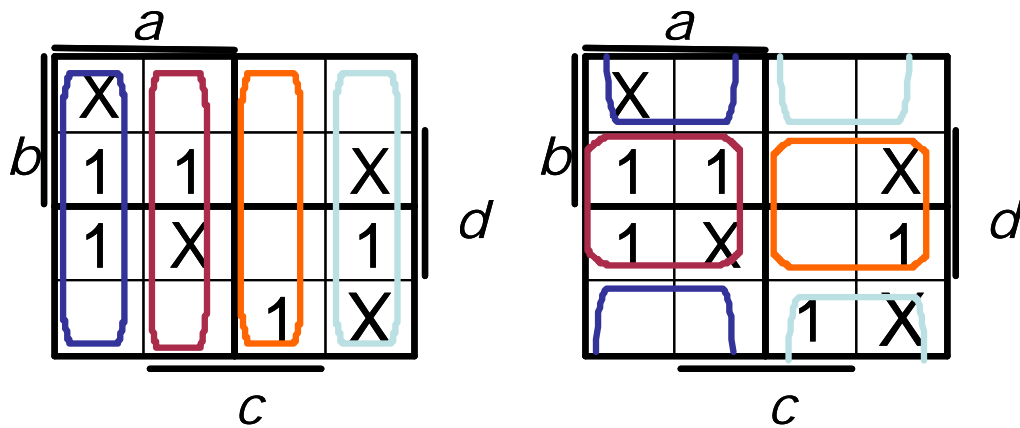
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

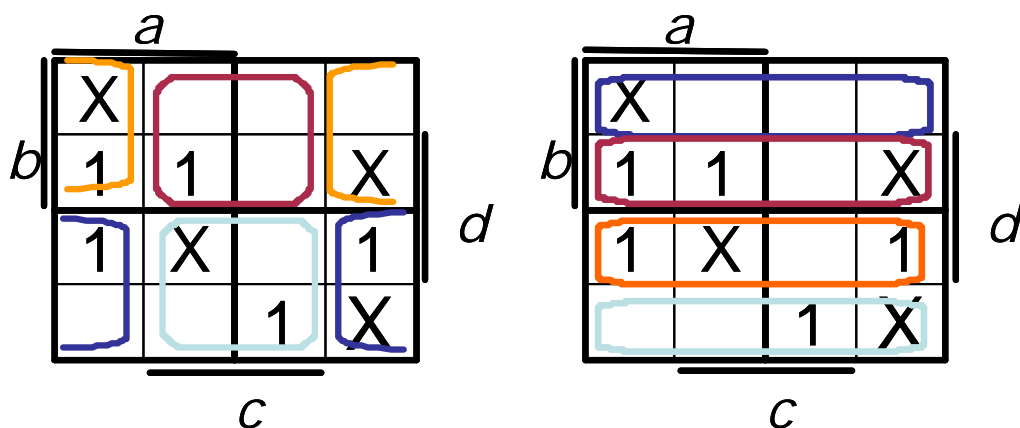
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

V zgornjem Veitch–evem diagramu so označena vsa štiri polja štirih mintermov, če izberemo vhodni spremenljivki  $a$  in  $b$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $ab="11"$ , oranžni kvadrat ko bo  $ab="01"$ , temno modri ko bo  $ab="10"$  in svetlo modri ko bo  $ab="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata opišemo s spremenljivko  $d$ , če postavimo redundanco na '0'. Vrednost spodnjega desnega kvadrata je bolj komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $c \cdot d$ , za spodnjo '1' pa  $c \cdot d'$ . Funkcija bo torej  $c \cdot d + c \cdot d'$ , kar je enačba funkcije XOR. Najbolj enostavna realizacija je zgornji desni kvadrat, ki je kar '0', če postavimo redundanco na '0'. Zato, da bi pregledali še ostale možnosti, moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

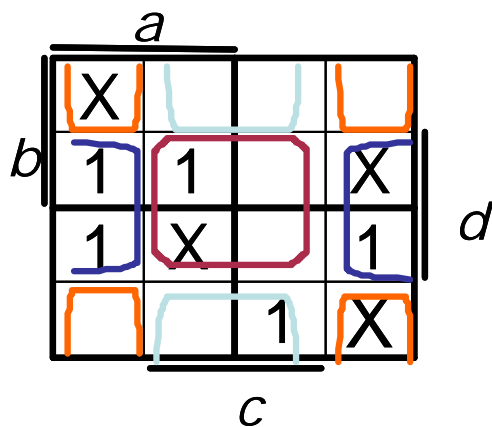
Če izberemo kot naslovni spremenljivki  $a$  in  $c$ , dobimo levi Veitchev diagram, če  $a$  in  $d$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najcenejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo inačice kvadratov, ki vsebujejo same '1' ali same '0'. Pri razvoju po  $a$  in  $c$  imamo pri  $ac="01"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'; medtem ko je razvoju po  $a$  in  $d$  nikjer ne nastopa ena sama '1' ali tri '1' ali diagonalna (XOR) dveh '1'.



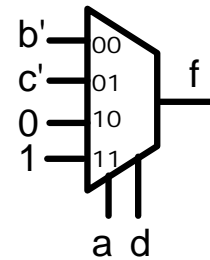
Nato izberemo naslovni spremenljivki  $b$  in  $c$ , (levi Veitchev diagram) in  $b$  in  $d$  (desni diagram). Pri razvoju po  $b$  in  $c$  imamo pri  $bc="11"$  najneugodnejšo funkcijo (rdeč), saj vsebuje eno samo '1'; medtem ko imamo pri razvoju po  $b$  in  $d$  pri  $bd="11"$  (rdeč) najneugodnejšo funkcijo, saj vsebuje tri '1'.



Zadnja kombinacija naslovnih vhodov je cd. Pri razvoju po c in d imamo pri cd="10" najneugodnejšo funkcijo (svetlo moder), saj vsebuje eno samo '1'. Najbolj ugodna kombinacija za realizacijo je torej razvoj po spremenljivkah a in d.



Končna realizacija funkcije:



Rešitev 4. naloge:

Število  $123_{10} = 01111011_2$ . Oziroma zapis posameznih števk: 0001 0010 0011 v BCD zapisu.

STOTICE				DESETICE				ENICE												
												0	1	1	1	1	0	1	1	START
										0		1	1	1	1	0	1	1		POMIK1
									0	1		1	1	1	0	1	1			POMIK2
									0	1	1	1	0	1	1					POMIK3
								0	1	1	1	1	0	1	1					ADD3
								1	0	1	0	1	0	1	1					POMIK4
							1	0	1	0	1	0	1	1						ADD3
							1	1	0	0	0	0	0	1	1					POMIK5
						1	1	0	0	0	0	0	1	1						POMIK6
					1	1	0	0	0	0	0	1	1							POMIK7
				1	0	0	1	0	0	0	1	1								ADD3
			1	0	0	1	0	0	0	1	1									POMIK8
1 <sub>10</sub>				2 <sub>10</sub>				3 <sub>10</sub>												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

# RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij  
24. 1. 2012

1. Prikažite postopek celoštevilskega deljenja nepredznačenih števil v dvojiškem sistemu na primeru:

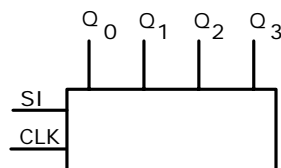
$$47_{10}/9_{10}$$

2. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2$  = konjunkcijo treh spremenljivk
- $f_3$  = funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh vseh.

Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko.

3. Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod *SI* (ang. serial input), in vzporedni izhod ( $Q_0, Q_1, Q_2, Q_3$ ).



4. Narišite tabelo prehajanja stanj Moore-ovega avtomata končnih stanj, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov. Krmilje ima:

- vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov,
- vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov,
- izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

Avtomat ne vrača drobiža in se ob detekciji plačila 15 centov ne vrača nazaj v začetno stanje, ampak ostane v končnem stanju. Vnos dveh kovancev naenkrat ni mogoč.

Rešitev 2. naloge:

Pretvorimo število v dvojiški zapis  $47_{10}=2F_{16}=0010\ 1111_2$

Podobno  $9_{10}=9_{16}=1001_2$

				0	0	1	0	1	kvocient=5 <sub>10</sub>					
1	0	0	1	0	0	1	0	1	1	1	1	1		START
				1	0	0	1							2 < 9
				0	0	1	0	1						
					1	0	0	1						5 < 9
					0	1	0	1	1					
						1	0	0	1					11>9→ odštejemo
						0	0	1	0	1				5 < 9
							1	0	0	1				
							0	1	0	1	1			11>9→ odštejemo
								1	0	0	1			
								0	0	1	0			ostanek=2 <sub>10</sub>

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



## Rešitev 2. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\begin{aligned} f_2 &= \overline{x_1 \cdot x_2 \cdot x_3} \\ \overline{f_2} &= \overline{x_1} + \overline{x_2} + \overline{x_3} \end{aligned}$$

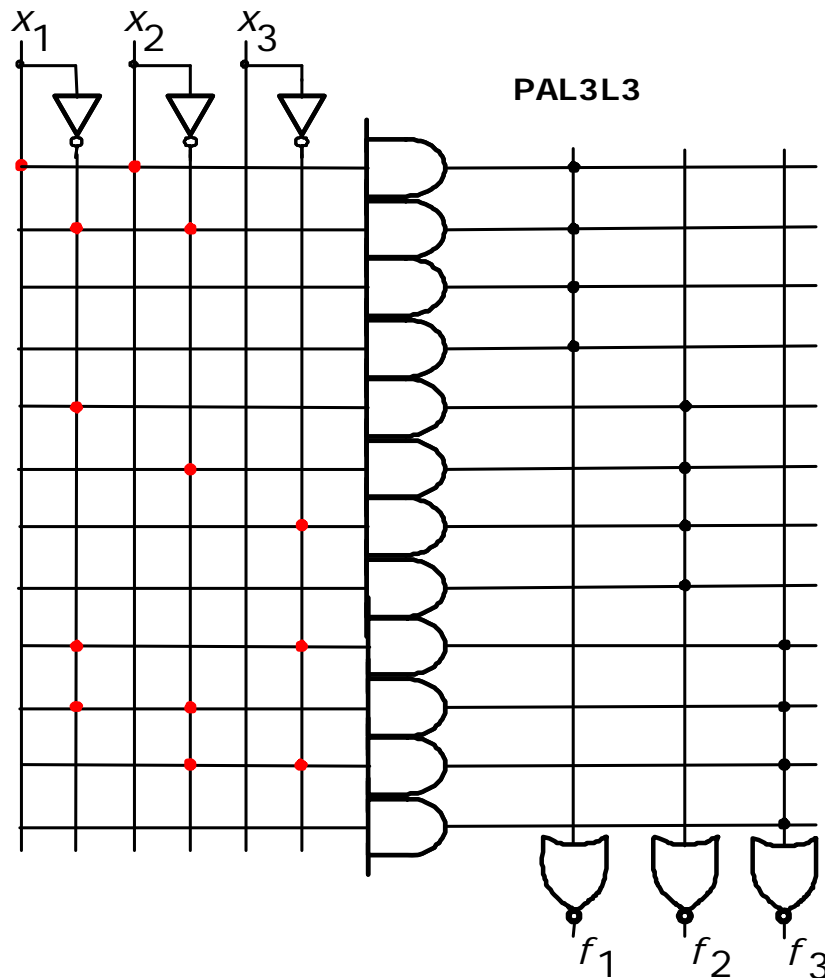
Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

$$\overline{f_3} = \overline{x_1} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_3}$$

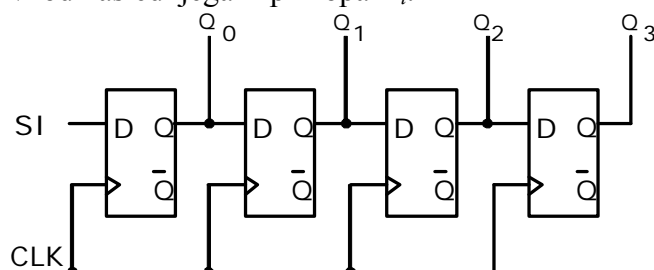
Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vežemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6-vhodna. Vezje PAL3L3 je AND-NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

### Rešitev 3. naloge:

Zaporedno–vzporedni (SIPO) pomikalni register, realiziran s pomočjo D–FF, je veriga kaskadno vezanih D–FF, v kateri je izhod prejšnjega flip–flopa  $Q_{i-1}$  vezan na vhod naslednjega flip–flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz T–FF in 2/1 izbiralnikov, moramo pravzaprav realizirati celico D–FF s pomočjo T–FF in 2/1 izbiralnikov. V ta namen zapišemo tabelo D–FF, pri kateri dodamo izhodni stolpec T vhoda.

$D$	$Q(t)$	$Q(t+1)$	$T$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Iz tabele sledi, da je  $T$  vhod XOR operacija  $Q(t)$  in vhoda D–FF, ki ga realiziramo.

$$Q(t+1) = Q(t) \oplus D$$

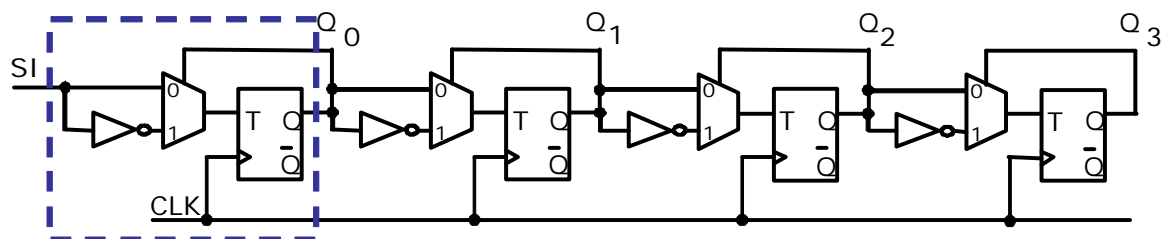
XOR vrata moramo realizirati s pomočjo 2/1 izbiralnikov, zato zapišemo enačbo XOR funkcije:

$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

Funkcijo  $f$  realiziramo z izbiralnikom tako, da naredimo razvoj po spremenljivki  $x$  in dobimo:

$x$	$f$
0	$y$
1	$y'$

Če nastali D–FF iz T–FF in 2/1 izbiralnika sestavimo skupaj v 4–bitni pomikalni register dobimo spodnjo realizacijo, v kateri je izvedba D–FF označena črtkano.



Opis delovanja in vezje pomikalnega registra je v predlogah avditornih vaj na domači strani predmeta: Logisim\shift\_reg\shift\_reg\_4bit\_using\_tff\_mux.circ

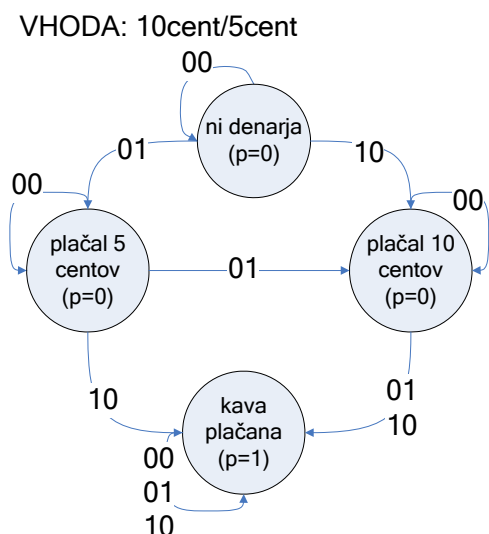
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 4. naloge:

Moore—ova realizacija avtomata končnih stanj. Opis diagrama stanj:



Na začetku se nahajamo v stanju "ni denarja", v katerem je izhod  $p=0$ . Vhoda v avtomat sta dva: 10cent in 5cent, kar na diagramu kodiramo kot 10cent/5cent.

Mehanizem za vnos kovancev preprečuje hkraten vnos dveh kovancev, torej je kombinacija (10cent/5cent=11) nemogoča, zato bo avtomat od tu lahko prešel v poljubno stanje (X). Če uporabnik ni vrgel denarja v avtomat (10cent/5cent=00), potem ostaja v stanju "ni denarja". Če uporabnik vrže v avtomat 5 centov (10cent/5cent=01), potem preide v stanje "plačal 5 centov". Če uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "plačal 10 centov". Ne glede na to koliko je

vrgel bo izhod v teh dveh stanjih enak  $p=0$ , ker še ni plačal celotne cene kave. Če smo v stanju "plačal 5 centov" in uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ). Stanje "kava plačana" je končno in tam tudi ostanemo za vse možne kombinacije. Če smo v stanju "plačal 10 centov" in uporabnik vrže v avtomat 5 ali 10 centov (10cent/5cent=10 oz. 01), potem podobno preidemo v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ).

Naredimo tabelo prehajanja stanj:

trenutno stanje	naslednje stanje (5cent/10cent)				izhod p
	00	01	10	11	
ni denarja	ni denarja	plačal 5 centov	plačal 10 centov	X	0
plačal 5 centov	plačal 5 centov	plačal 10 centov	kava plačana	X	0
plačal 10 centov	plačal 10 centov	kava plačana	kava plačana	X	0
kava plačana	kava plačana	kava plačana	kava plačana	X	1

Celotno sintezo avtomata si lahko ogledate v zbirki vaj na domači strani predmeta.

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
07. 12. 2012

1. Z uporabo pravil Boole-ove logike zapišite podano logično funkcijo z NAND (Sheffer-jevimi) operatorji.

$$f(x_1, x_2, x_3) = (x_1 + x_2) \cdot (\overline{x_2} + \overline{x_3}) \cdot x_3$$

2. Določite redundance podane funkcije  $f$  tako, da bo nastala funkcija linearna in izračunajte koeficiente linearnosti.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5-7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

4. Pretvorite število  $197_{10}$  ( $11000101_2$ ) v BCD zapis z uporabo "double dabble" algoritma.

## Rešitev 1. naloge

Z uporabo pravil Boole-ove logike moramo zapisati podano logično funkcijo z NAND (Sheffer-jevimi) operatorji. Podana funkcija je v obliki KNO.

Direktne poti za pretvorbo KNO v zapis z NAND (Sheffer-jevimi) operatorji ni, zato KNO najprej pretvorimo v DNO obliko, tako da izvedemo AND operacije nad dvočleniki:

$$f(x_1, x_2, x_3) = (x_1 + x_2) \cdot (\overline{x_2} + \overline{x_3}) \cdot x_3$$

V drugem členu bomo dobili člen  $x_2 \cdot x_2' = 0$  po lastnostih Boole-ove algebre:

$$f(x_1, x_2, x_3) = (x_1 \cdot \overline{x_2} + x_1 \cdot \overline{x_3} + x_2 \cdot \overline{x_3}) \cdot x_3$$

S preostalimi členi izpišemo AND operacije:

$$f(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_3} \cdot x_3 + x_2 \cdot \overline{x_3} \cdot x_3$$

Podobno pri drugem in tretjem členu funkcije dobimo  $x_3 \cdot x_3' = 0$ :

$$f(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} \cdot x_3$$

Nastala funkcija ima samo en člen, ki ga moramo izraziti z NAND operatorji. Funkcijo negiramo in dobimo:

$$\overline{f(x_1, x_2, x_3)} = \overline{x_1 \cdot \overline{x_2} \cdot x_3}$$

Izrazimo še negacijo nekega izraza  $x$  s pomočjo NAND operatorjev:

$$\overline{x} = 1 \uparrow x$$

$$\overline{x} = x \uparrow x$$

Nazadnje izrazimo original funkcije  $f$ , kot zahteva naloga:

$$f(x_1, x_2, x_3) = 1 \uparrow (x_1 \uparrow (1 \uparrow x_2) \uparrow x_3)$$

Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:

	$x_1$			
$x_2$	1		1	
		X		1
	1		X	
		1		1
			$x_3$	$x_4$

Če bo funkcija linearna, jo bomo lahko realizirali s pomočjo XOR funkcij. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.

	$x_1$			
$x_2$	1		1	
		X		1
	1		X	
		1		1
			$x_3$	$x_4$

Diagram is annotated with XOR operations for flipping:

- Top-left to top-right:  $k_0 \oplus k_1 \oplus k_2$
- Top-right to top-left:  $k_0 \oplus k_2$
- Bottom-right to top-right:  $k_0 \oplus k_2 \oplus k_4$
- Bottom-right to bottom-left:  $k_0$
- Bottom-left to bottom-right:  $k_0 \oplus k_3$

Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

### Rešitev 3. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

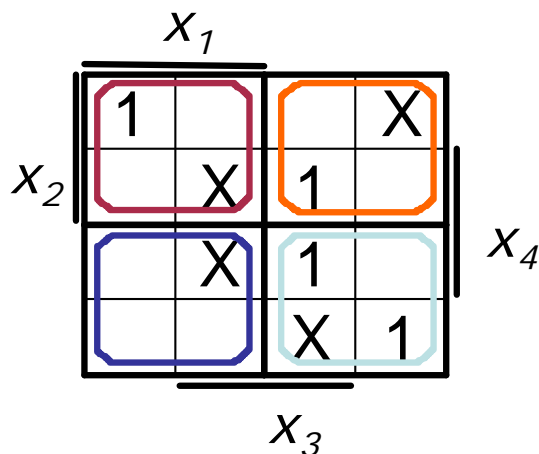
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

Dobljeno funkcijo vrišemo v Veitch–ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev–em diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:

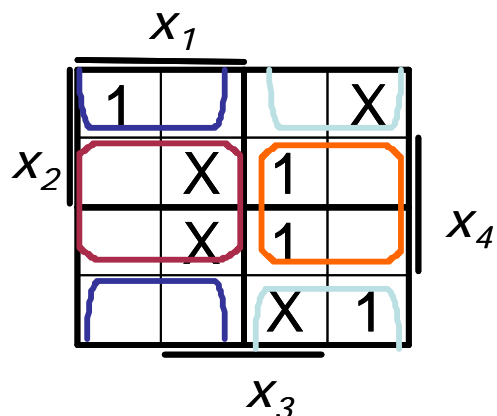
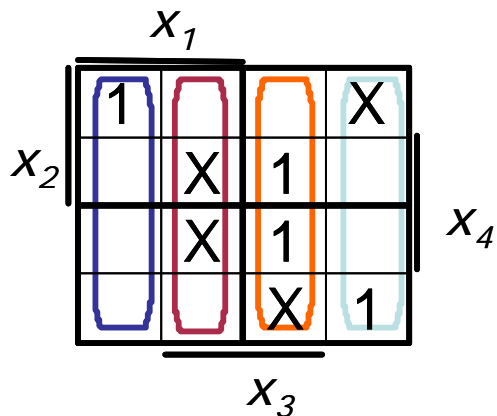
$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X



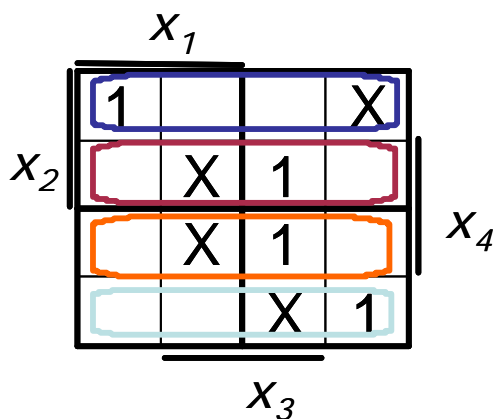
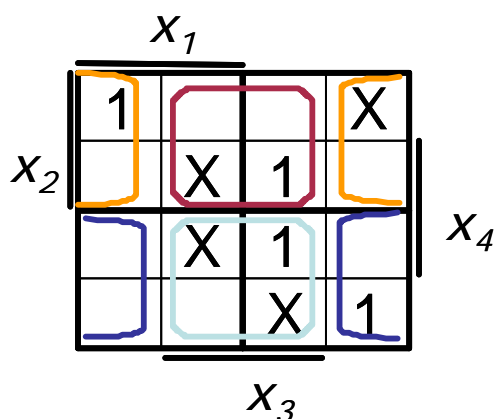
V zgornjem Veitchevem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2="11"$ , oranžni kvadrat ko bo  $x_1x_2="01"$ , temno modri ko bo  $x_1x_2="10"$  in svetlo modri ko bo  $x_1x_2="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3x_4$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4$ . Funkcija bo torej  $x_3x_4 + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov. Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram, če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo drugače kvadratov, ki vsebujejo konstante (samo '1' ali samo '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

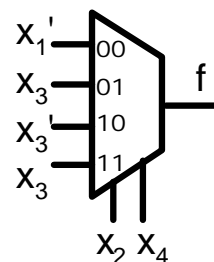
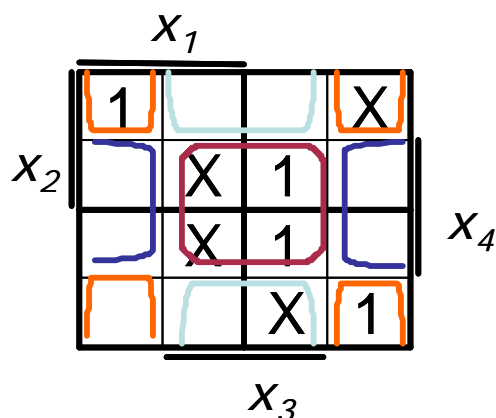




Nato izberemo naslovni spremenljivki  $x_2$  in  $x_3$ , (levi diagram) in  $x_2$  in  $x_4$  (desni diagram). Pri razvoju po  $x_2$  in  $x_3$  imamo pri  $x_2x_3="00"$  najneugodnejšo funkcijo (moder), saj vsebuje eno samo '1'. Pri razvoju po  $x_2$  in  $x_4$  nikjer nimamo osamljene '1', zato se dajo funkcijski ostanki enostavno realizirati, če vse redundance postavimo na '1'.



Zadnja kombinacija naslovnih vhodov je  $x_3$  in  $x_4$ . Pri razvoju po  $x_3$  in  $x_4$  imamo pri  $x_3x_4="00"$  najneugodnejšo funkcijo (oranžen), saj dve '1' opišemo s funkcijo ekvivalence. Možno rešitev torej predstavlja kombinacija naslovnih vhodov  $x_2$  in  $x_4$ .



Vezje izbiralnika je v predlogah avditornih vaj na domači strani predmeta:  
Logisim\MUX\mux\_4\_1\_f\_V\_0\_3\_7\_12\_in\_Vx\_2\_4\_11\_15.circ

Rešitev 4. naloge:

Število  $197_{10} = 11000101_2$ . Zapis posameznih števk v BCD zapisu se glasi: 0001 1001 0111.

STOTICE				DESETICE				ENICE												
												1	1	0	0	0	1	0	1	START
										1		1	0	0	0	1	0	1		POMIK1
									1	1		0	0	0	1	0	1			POMIK2
									1	1	0	0	0	1	0	1				POMIK3
								1	0	0	1	0	0	1	0	1				ADD3
							1	0	0	1	0	0	1	0	1					POMIK4
						1	0	0	1	0	0	1	0	1						POMIK5
					1	0	0	1	0	0	1	0	1							ADD3
					1	0	0	1	1	0	0	0	1							POMIK6
				1	0	0	1	1	0	0	0	1								ADD3
				1	1	0	0	1	0	1	1	1								POMIK7
			1	1	0	0	1	0	1	1	1									POMIK8
$1_{10}$				$9_{10}$				$7_{10}$												

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

# RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij

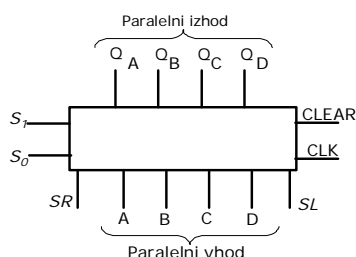
18. 1. 2013

1. Uporabite PAL3L3 (namišljen čip) za realizacijo naslednjih funkcij:

- $f_1 = x_1 \oplus x_2$
- $f_2 = x_1 \cdot x_2 \cdot x_3$
- $f_3 =$  funkcijo treh spremenljivk, ki vrne '1' pri vsaj dveh enicah na vseh.

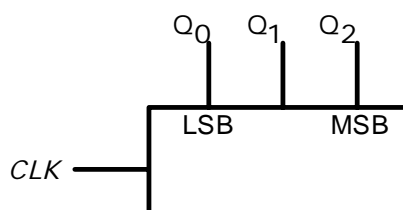
Vezje ima 3 vhode in 3 izhode. Vsaka disjunkcija (OR) ima 4 konjunkcije (AND). Oznaka L pomeni, da je izhod negiran. Programirane povezave označite s piko. Vezje PAL je narisano na hrbtni strani kolokvija.

2. Z uporabo D flip-flopov in izbiralnikov 4/1 prikažite sintezo univerzalnega 4-bitnega pomikalnega registra, ki ima dva funkcijska vhoda  $S_0$  in  $S_1$  in opravlja funkcije po spodnji tabeli. Register ima tudi zaporedna vhoda za pomik v levo (SL – serial left), pomik v desno (SR – serial right) in asinhroni vhod za brisanje CLEAR (aktiven nizek).



$S_1$	$S_0$	<i>funkcija</i>
0	0	drži stanje
0	1	pomik vsebine eno mesto desno
1	0	pomik vsebine eno mesto levo
1	1	nalaga vsebino z vhodov ABCD

3. Prikažite sintezo sinhronnega 3-bitnega števca navzdol z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov ter vezje narišite. Imena signalov so razvidna iz spodnje slike.



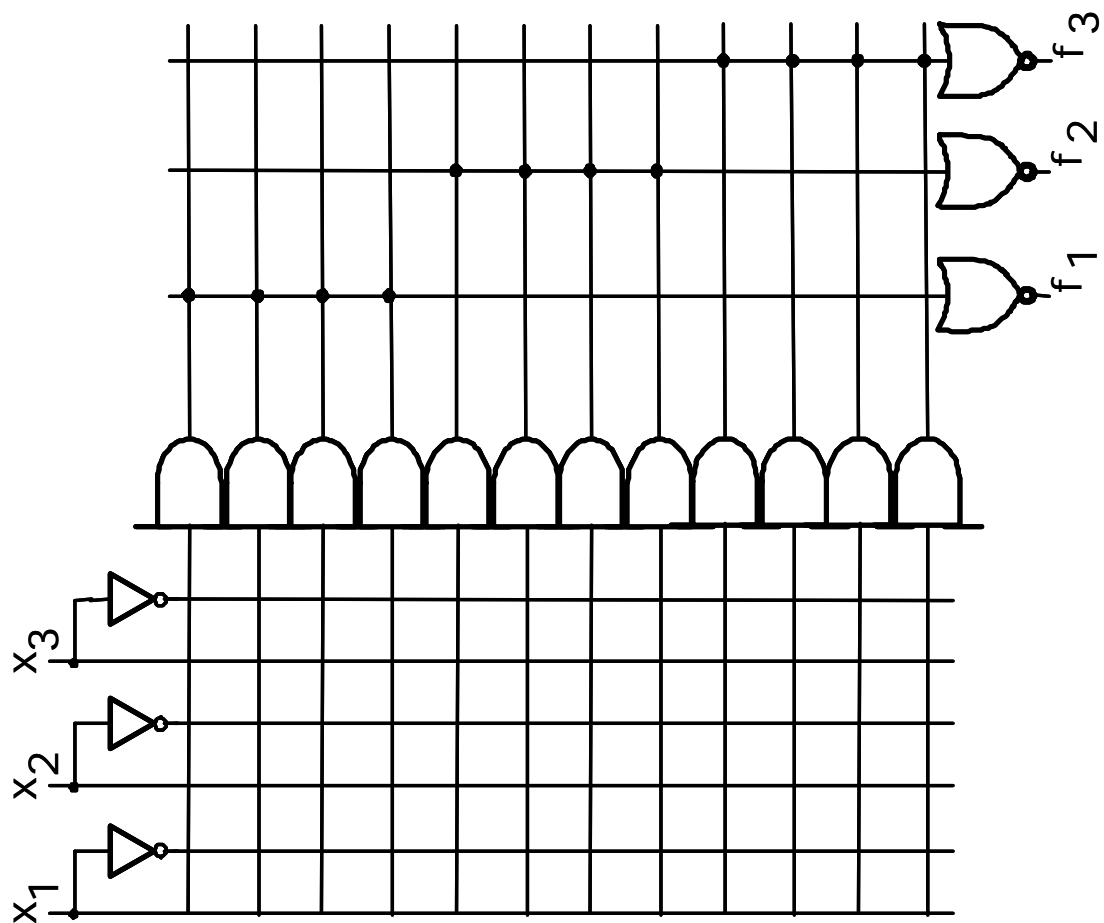
4. Narišite diagram stanj za avtomat končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z=1$ , ko se na vhodu pojavi zaporedje **110** ali **101**, sicer je  $z=0$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda. Tip avtomata je razviden iz podanega časovnega zaporedja.

$CLK$	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
$w$	0	1	1	0	1	1	0	1	1	1	0	0	0
$z$	–	0	0	1	1	0	1	1	0	0	1	0	0

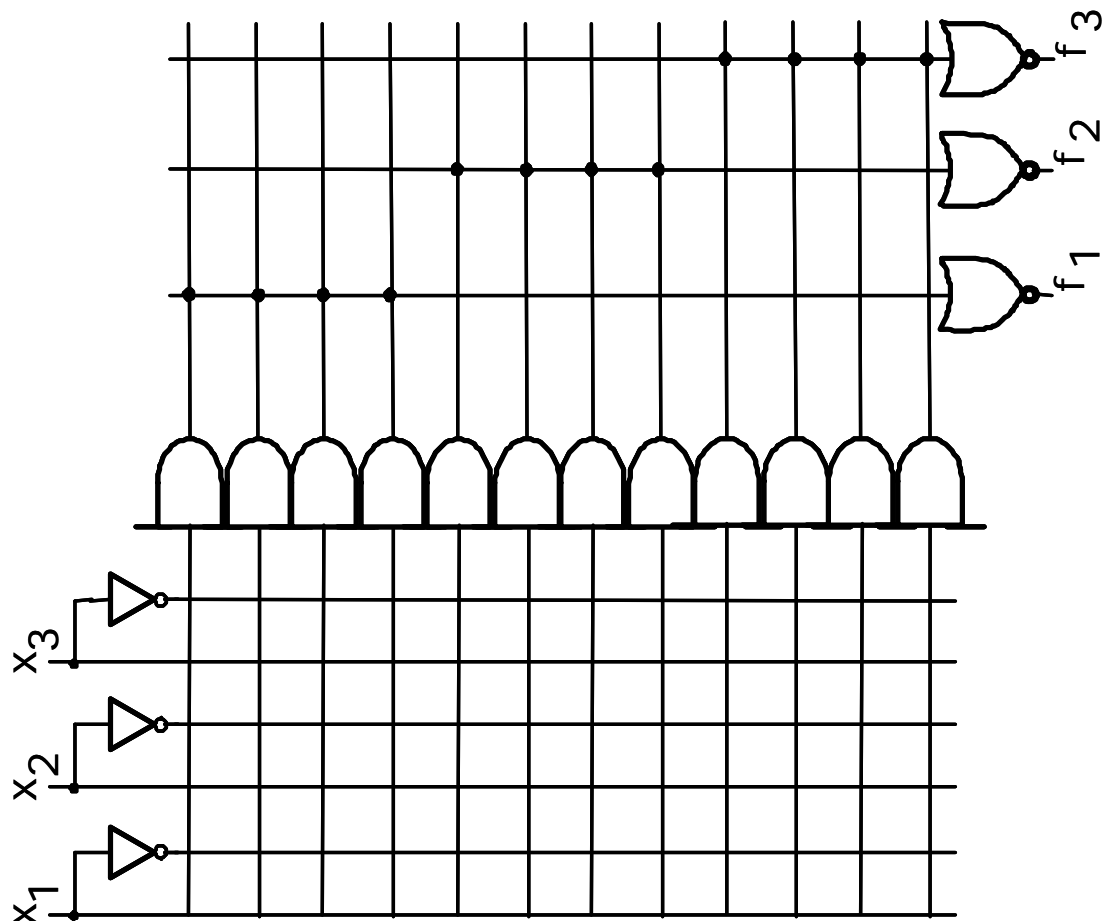
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.  
 Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.  
 Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# Rešitev 1. naloge:

Za funkcije zapišemo najprej pravilnostno tabelo, nato narišemo Veitch–eve diagrame.

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Vezje PAL ima negirane izhode, zato bomo pri realizaciji funkcij z Veitch–evimi diagrami realizirali  $\overline{f}$  in ne  $f$ .

$\overline{f_1}$ :

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

Prvo funkcijo zapišemo enostavno, saj je negacija XOR funkcije dveh spremenljivk kar funkcija ekvivalence:

$$\overline{f_1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$$

Podobno lahko naredimo za drugo funkcijo, kjer za negacijo konjunkcije treh spremenljivk uporabimo De Morgan–ovo enakost.

$\overline{f_2}$ :

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

$$\overline{f_2} = \overline{x_1 \cdot x_2 \cdot x_3}$$

$$\overline{f_2} = \overline{x_1} + \overline{x_2} + \overline{x_3}$$

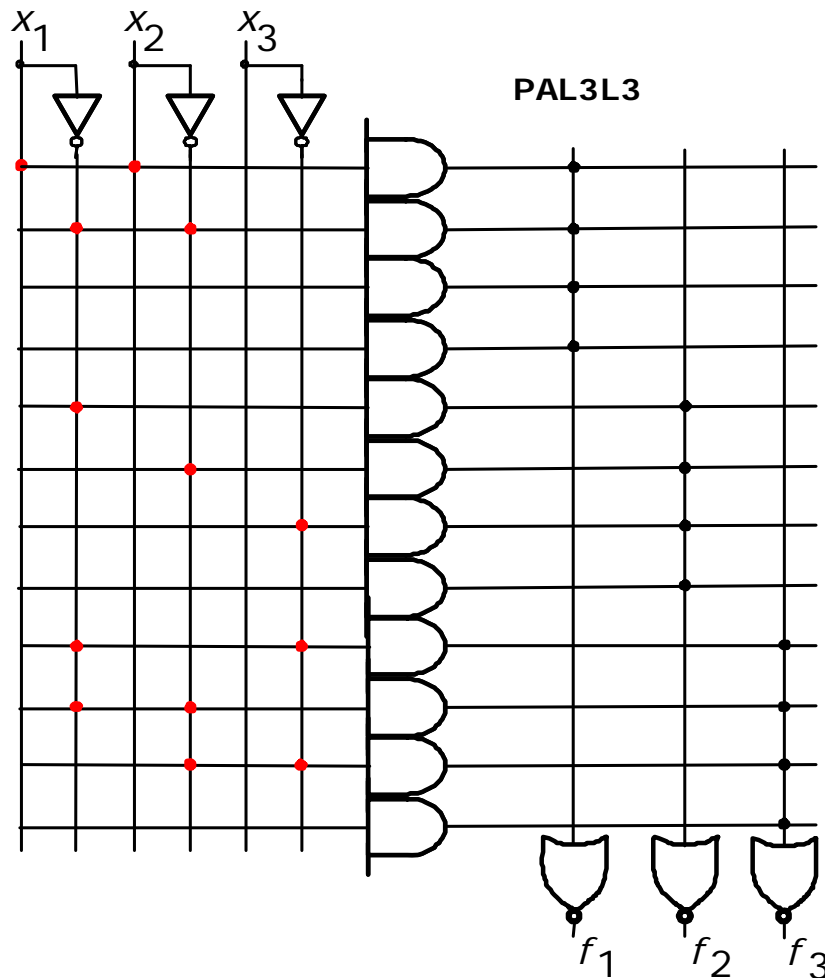
Zadnjo funkcijo minimiziramo z uporabo Veitch–evega diagrama, tako da zbiramo ničle.

$\overline{f_3}$ :

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

$$\overline{f_3} = \overline{x_1 \cdot x_3} + \overline{x_1 \cdot x_2} + \overline{x_2 \cdot x_3}$$

Pri realizaciji PAL vezja upoštevamo poenostavljeno strukturo, pri kateri ne vežemo vsake povezave na konjunkcije, saj so AND vrata na narisani strukturi 6-vhodna. Vezje PAL3L3 je AND-NOR arhitekture in vsebuje 4 konjunkcije na en NOR člen. Pri PAL vezju je programabilen samo AND del vezja.



Predstavljeno PAL3L3 vezje je sicer izmišljeno, vendar demonstrira strukturo in uporabo večjih (realnih) PAL vezij kot so npr. PAL14L4, GAL16V8 in GAL22V10. GAL vezja so nadgradnja osnovne PAL strukture. Slednji so izključno kombinacijski, GAL vezja pa imajo v OLMC (ang. Output Logic MacroCell) strukturi še D-FF, s katerim lahko realiziramo tudi sekvenčna vezja.

## Rešitev 2. naloge:

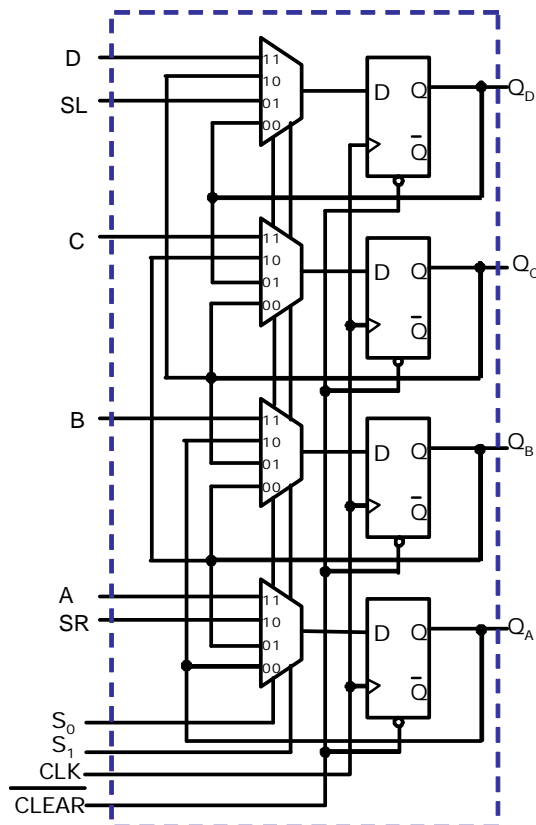
Vsako od operacij izpišemo v pravilnostno tabelo v kateri združimo funkcijska bita  $S_1$  in  $S_0$  in trenutno stanje na  $i$ -tem mestu registra  $Q_i(t)$ . Mesta registra od leve proti desni so  $Q_i = (Q_A, Q_B, Q_C, Q_D)$ . Realizacija z D flip-flopi nam analizo močno poenostavi, zaradi enačbe D flip-flopa:  $D = Q(t+1)$ .

Tabela 1: Prehajanje stanj univerzalnega registra.

$S_1$	$S_0$	$Q_i(t+1)$	funkcija
0	0	$Q_i(t)$	HOLD
0	1	$Q_{i+1}(t)$	LSR
1	0	$Q_{i-1}(t)$	LSL
1	1	$x_i$	LOAD

Iz poenostavljene tabele prehajanja stanj univerzalnega registra sestavimo realizacijo, ki bo vključevala izbiralnike MUX 4/1 in D-FF.

Na naslovna vhoda vseh MUX 4/1 vodimo funkcijska signala  $S_1$  in  $S_0$ . Potem na vsakem podatkovnem vhodu realiziramo ustrezno funkcijo.



Stanje  $S_1S_0="00"$  pomeni držanje stanja (*HOLD*), torej bodo trenutne vrednosti D-FF ohranile vrednost  $Q_i(t+1) = Q_i(t)$ . Na sliki to realiziramo tako, da vodimo izhod D-FF nazaj na vhod pri podatkovnem vhodu 00. Stanje  $S_1S_0="01"$  pomeni pomik desno (*LSR* – ang. logic shift right), torej bodo D-FF pomaknili vsebino eno mesto desno. Pomik desno pomeni, da na mesto skrajno levega bita vpišemo vrednost zaporednega vhoda SR, nato  $Q_A$  vodimo na vhod  $Q_B$  in tako do skrajno desnega bita. Stanje  $S_1S_0="10"$  pomeni pomik levo (*SHL* – ang. shift left), torej bodo D-FF pomaknili vsebino eno mesto levo. Pomik levo pomeni, da na mesto skrajno desnega bita vpišemo vrednost zaporednega vhoda SL, nato  $Q_D$  vodimo na vhod  $Q_C$  in tako do skrajno levega bita.  $S_1S_0="11"$  pomeni vzporedno nalaganje z vhodov (*LOAD*)  $Q_D(t+1)=D$ ,  $Q_C(t+1)=C$ ,  $Q_B(t+1)=B$ ,  $Q_A(t+1)=A$ . Na tabeli smo  $i$ -ti vhod za vzporedno nalaganje označili kot  $x_i = (A, B, C, D)$ .

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števec:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch-eve diagrame za tri spremenljivke za vsak vhod T-FF, vendar ker so T-FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Iz stolpca T<sub>0</sub> se vidi:

$$T_0 = 1$$

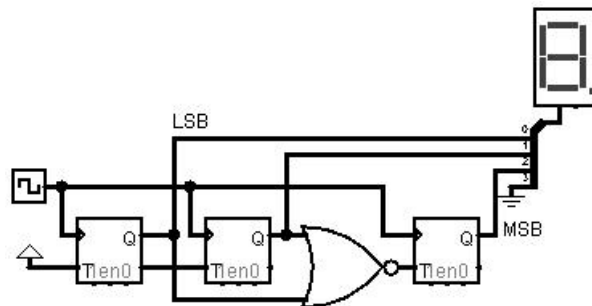
Z opazovanjem stolpcev trenutnega stanja določimo T<sub>1</sub>:

$$T_1 = \overline{Q_0}$$

Podobno lahko določimo T<sub>2</sub>:

$$T_2 = \overline{Q_0} \cdot \overline{Q_1} = \overline{Q_0 + Q_1}$$

Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_7\_0\_using\_T\_FF.circ:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

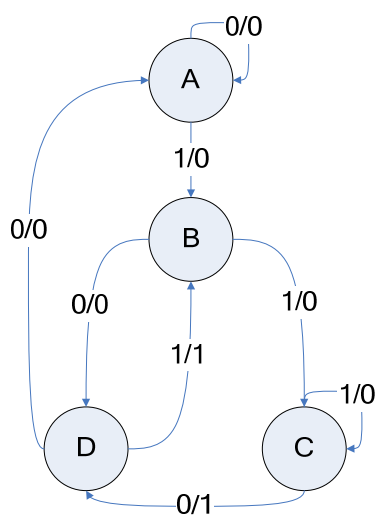
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



#### Rešitev 4. naloge:

Naloga zahteva realizacijo z Mealy–evim tipom avtomata. Zapišemo začetno stanje A, v katerem ostajamo toliko časa, dokler se ne začne ena od sekvenc, ki ju zaznavamo. Obe sekvenci se začneta z '1', zato v stanje B preidemo, ko je na vhodu prva '1'. V stanju B ne moremo ostati, saj se na vhodu lahko pojavi '0' ali '1' – v obeh primerih gre za del zaznavanega zaporedja "10X" ali "11X". Iz stanja B preidemo v stanje C, če se vmes pojavi '1', tako da v tem stanju pomeni detekcijo sekvence "11X", v stanje D pa preidemo če se pojavi na vhodu '0', kar pomeni detekcijo sekvence "10X".

Prekrivanje zaporedij: Če se v stanju C pojavi '1' na vhodu, potem gre za sekvenco "111" na vhodu – kar še vedno pomeni, da ostajamo v stanju C, saj je prekrivanje vzorcev dovoljeno. Drugače se diagram obnaša, ko smo v stanju D in pride na vhod še ena '0' – takrat smo imeli na vhodu sekvenco "100", tako da se moramo vrniti v stanje A, saj se nobena od zaznavanih sekvenc ne začneja z '0'.



Delovanje avtomata preizkusimo na testnem zaporedju:

stanje	A	B	D	B	D	A	B	C	D	B	D	A	B	C	D	B	D	B	C	D	A	B	C	D
w	0	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	1	1	0	0	1	1	0
z	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
12. 12. 2013

1. Določite minimalno normalno obliko (MNO) za logično funkcijo  $f$  z redundantnimi vhodnimi kombinacijami. Izbiro funkcije utemeljite z izračunom  $COST$  funkcije.

$$f^4 = \sum (0,1,6,7,14,15) \text{ in } \sum_x (3,5,9,11,13)$$

2. Z uporabo "carry lookahead" načina seštevanja seštejte števili  $x$  in  $y$ . Na vseh mestih določite vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ). Končni rezultat izračunajte z uporabo prenosov ( $c_i$ ), ki izvirajo iz pomočjo funkcij ( $g_i$ ) in ( $p_i$ ), ter vhodnih operandov  $x$  in  $y$ .

$$x + y = 19_{10} + 23_{10} = 42_{10}$$

3. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1,2,5-7,9,10,14) \text{ in } \&_x(0,4,11,13)$$

4. Pretvorite število  $293_{10}$  ( $100100101_2$ ) v BCD zapis z uporabo "double dabble" algoritma.

# DEVELOPMENT OF DIGITAL SYSTEMS

Midterm Examination

12. 12. 2013

1. Determine the minimal normal form (MNO) of a given incompletely specified function  $f$ . Write the minimal POS (product-of-sums) and SOP (sum-of-products) form, calculate the COST function of each one and determine MNO based on calculated COST function.

$$f^4 = \sum (0,1,6,7,14,15) \text{ in } \sum_x (3,5,9,11,13)$$

2. Add two unsigned numbers  $x$  and  $y$  using carry look-ahead addition technique. Evaluate generate ( $g_i$ ) and propagate ( $p_i$ ) functions for all bit positions. Calculate the sum ( $S_i$ ) using carry bits ( $c_i$ ) and both inputs ( $x_i, y_i$ ). Carry bits ( $c_i$ ) have to be evaluated using generate ( $g_i$ ) and propagate ( $p_i$ ) functions.

$$x + y = 19_{10} + 23_{10} = 42_{10}$$

3. Implement the incompletely specified function  $f$  in POS (product-of-sums) form using a single 4/1 multiplexer using Shannon expansion theorem.

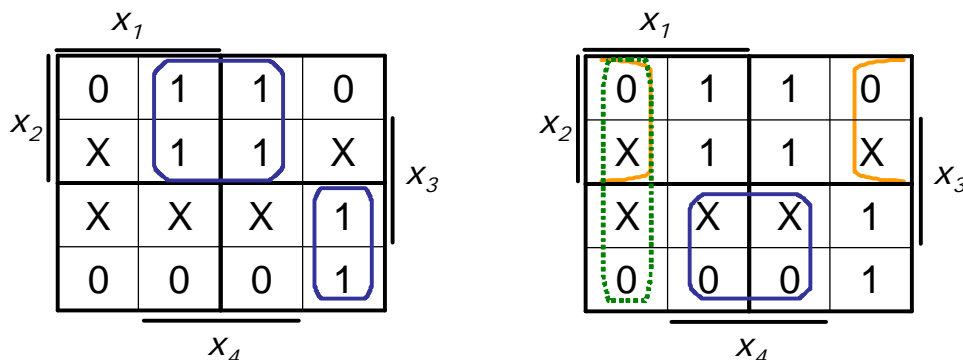
$$f(x_1, x_2, x_3, x_4) = \&(1,2,5-7,9,10,14) \text{ in } \&_x(0,4,11,13)$$

4. Convert the number  $293_{10}$  ( $100100101_2$ ) into BCD representation using "double dabble" algorithm.

Rešitev 1. naloge:

Funkcijo v PDNO z redundancami moramo izpisati v Veitch–ev diagram, od koder bomo lahko izvajali postopek minimizacije.

$$f^4 = \sum (0, 1, 6, 7, 14, 15) \text{ in } \sum_x (3, 5, 9, 11, 13)$$



Prikazana sta dva Veitch–eva diagrama. Levega uporabljamo za tvorbo MDNO, saj prikazuje funkcijo  $f$ , desnega za tvorbo MKNO, ker združujemo minterme negirane funkcije.

MDNO:

$$f_{MDNO} = x_2 \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_4}$$

Za MKNO pa združujemo ničle (izražamo negirano funkcijo) in izrazimo minimalno obliko negirane funkcije. Nato uporabimo De Morgan–ov teorem, da pridemo do konjunktivne izražave.

$$\begin{aligned} \overline{f} &= x_1 \cdot \overline{x_4} + \overline{x_2} \cdot x_4 + x_2 \cdot \overline{x_4} \\ f &= \overline{x_1 \cdot \overline{x_4} + \overline{x_2} \cdot x_4 + x_2 \cdot \overline{x_4}} \\ f &= \overline{x_1 + x_4} \cdot \overline{x_2 + x_4} \cdot \overline{x_2 + x_4} \\ f_{MKNO} &= (\overline{x_1 + x_4}) \cdot (\overline{x_2 + x_4}) \cdot (\overline{x_2 + x_4}) \end{aligned}$$

Za minimalno normalno obliko moramo določiti, katera izvedba funkcije je cenejša (manjši COST vezja).

OBLIKA	VRAT	VHODOV	COST
MDNO	3	7	10
MKNO	4	9	13

Cenejša izvedba je MDNO, saj je strošek vezja manjši (COST), zato je MNO=MDNO.

## Rešitev 2. naloge:

Uporaba "carry look ahead" načina seštevanja zahteva določitev vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ). Končni rezultat izračunamo z uporabo prenosov, ki izvirajo iz pomožjo funkcij ( $g_i$ ) in ( $p_i$ ), ter vhodnih operandov  $x$  in  $y$ . Bistvena prednost tega načina seštevanja je, da prenose lahko izračunamo vnaprej (ang. look ahead) vzporedno na osnovi enačb, s katerimi dani prenos izrazimo kot funkcijo ( $g_i$ ) in ( $p_i$ ) ter vhodnega prenosa. Zaradi vzporednega načina izračuna prenosov je izračun vsote neodvisen od števila mest seštevalnika. Problem tega načina so kompleksne funkcije, ki nastanejo pri naraščajočem številu mest seštevanja, zato se CLA seštevalniki združujejo po 4 mesta, izhodi za funkcije tvorjenja ( $g$ ) in širjenja ( $p$ ) teh CLA pa se vežejo na generator izhodnih prenosov (ang. carry look ahead generator oz. CLAG).

Način seštevanja "carry look ahead" zahteva določitev vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ) ter prenosa na naslednje mesto ( $c_{i+1}$ ) za vsako mesto posebej.

Osnovne enačbe "carry look ahead" so:

$$\begin{aligned}g_i &= x_i \cdot y_i \\p_i &= x_i \oplus y_i \\c_{i+1} &= g_i + p_i \cdot c_i\end{aligned}$$

Z iterativnim vstavljanjem formule za prenos  $c_{i+1}$  od mesta 0 do mesta 5 (mesto izhodnega prenosa) dobimo:

$$\begin{aligned}c_1 &= g_0 + p_0 \cdot c_{in} \\c_2 &= g_1 + p_1 \cdot c_1 = g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in}) \\c_3 &= g_2 + p_2 \cdot c_2 = g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in})) \\c_4 &= g_3 + p_3 \cdot c_3 = g_3 + p_3 \cdot (g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in}))) \\c_5 &= g_4 + p_4 \cdot c_4 = g_4 + p_4 \cdot (g_3 + p_3 \cdot (g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in}))))\end{aligned}$$

Pri seštevanju upoštevamo, da vhodnega prenosa ni ( $c_{in}=0$ ). Preostali prenosi se izračunajo vzporedno iz izrazov iterativnega vstavljanja, kar je bistvena prednost "carry look ahead" načina seštevanja pred običajnim "ripple carry" načinom seštevanja.

$$\begin{aligned}c_0 &= c_{in} = 0 \\c_1 &= g_0 + p_0 \cdot c_{in} = 1 + 0 \cdot 0 = 1 \\c_2 &= g_1 + p_1 \cdot c_1 = g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in}) = 1 + 0 \cdot 1 = 1 \\c_3 &= g_2 + p_2 \cdot c_2 = g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in})) = 0 + 1 \cdot 1 = 1 \\c_4 &= g_3 + p_3 \cdot c_3 = g_3 + p_3 \cdot (g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in}))) = 0 + 0 \cdot 1 = 0 \\c_5 &= g_4 + p_4 \cdot c_4 = g_4 + p_4 \cdot (g_3 + p_3 \cdot (g_2 + p_2 \cdot (g_1 + p_1 \cdot (g_0 + p_0 \cdot c_{in})))) = 1 + 0 \cdot 0 = 1\end{aligned}$$

$i$	5	4	3	2	1	0
$x_i$	0	1	0	0	1	1
+	$y_i$	0	1	0	1	1
<hr/>						
	$g_i$	0	1	0	0	1
	$p_i$	0	0	0	1	0
	$c_i$	1	0	1	1	0
<hr/>						
	$S_i$	1	0	1	0	1

Mesta vsote  $S_i$  dobimo z XOR operacijo  $S_i = x_i \oplus y_i \oplus c_i$  pri čemer XOR predstavlja vsoto po modulu 2 oz. funkcijo v PDNO:  $S_i(x_i, y_i, c_i) = V(1, 2, 4, 7)$ .

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

### Rešitev 3. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

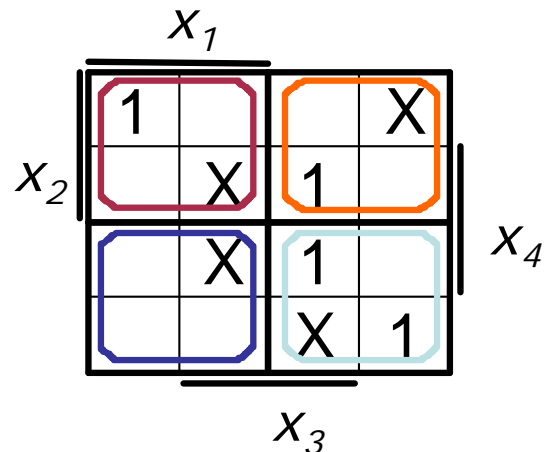
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

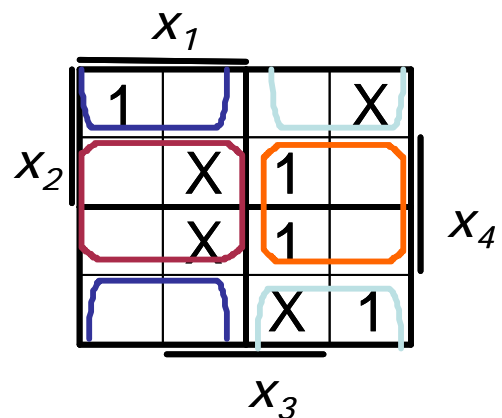
Dobljeno funkcijo vrišemo v Veitch–ev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchev–em diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X



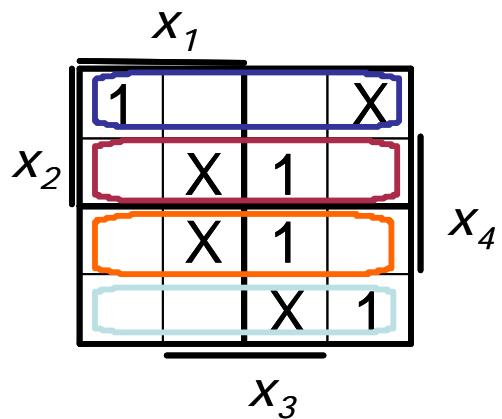
V zgornjem Veitchevem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2="11"$ , oranžni kvadrat ko bo  $x_1x_2="01"$ , temno modri ko bo  $x_1x_2="10"$  in svetlo modri ko bo  $x_1x_2="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimbolj enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3x_4$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4$ . Funkcija bo torej  $x_3x_4 + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov. Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram, če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo drugače kvadratov, ki vsebujejo konstante (same '1' ali same '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

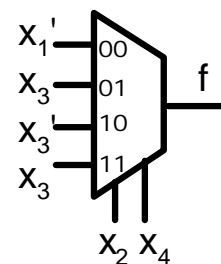


A 4x4 grid representing a 2D lattice with four input variables:  $x_1$  (top),  $x_2$  (left),  $x_3$  (bottom), and  $x_4$  (right). The grid contains binary values (0 or 1) and 'X' marks. Colored outlines highlight specific paths: an orange path (top-left to top-right), a purple path (bottom-left to bottom-right), a green path (middle-left to middle-right), and a red path (top-middle to bottom-middle).

	$x_1$				
$x_2$	1	X	1	X	
	X	1	X	1	
	X	1	X	1	
	X	1	X	1	
	$x_3$				
					$x_4$



A 4x4 grid representing a 2D lattice. The four boundary variables are labeled:  $X_1$  (top),  $X_2$  (left),  $X_3$  (bottom), and  $X_4$  (right). The grid contains values 1 and X. The top row has 1, empty, empty, X. The second row has empty, X, 1, empty. The third row has empty, X, 1, empty. The bottom row has empty, empty, X, 1. Colored loops (orange, blue, red) connect the boundary variables to the internal values.



**Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>**

Rešitev 4. naloge:

Število  $293_{10}$  pretvorimo v dvojiško obliko:  $1\ 0010\ 0101_2$ . Zapis posameznih števk v BCD zapisu se glasi: 0010 1001 0011. Število  $293_{10}$  je 9 bitno število, zato je pomikov 9.

STOTICE	DESETICE	ENICE	1	0	0	1	0	0	1	0	1	START
		1	0	0	1	0	0	1	0	1		POMIK1
		1 0	0	1	0	0	1	0	1			POMIK2
		1 0 0	1	0	0	1	0	1				POMIK3
		1 0 0 1	0	0	1	0	1					POMIK4
		1 1 0 0	0	0	1	0	1					ADD3
	1	1 0 0 0	0	1	0	1						POMIK5
	1	1 0 1 1	0	1	0	1						ADD3
	1 1	0 1 1 0	1	0	1							POMIK6
	1 1	1 0 0 1	1	0	1							ADD3
	1 1 1	0 0 1 1	0	1								POMIK7
	1 0 1 0	0 0 1 1	0	1								ADD3
1	0 1 0 0	0 1 1 0	1									POMIK8
1	0 1 0 0	1 0 0 1	1									ADD3
1 0	1 0 0 1	0 0 1 1										POMIK9
$2_{10}$	$9_{10}$	$3_{10}$										



# RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij  
16. 1. 2014

1. Uporabite ROM vezje za realizacijo naslednjih funkcij:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

ROM vezje ima 3 vhodne spremenljivke in 4 bitno vsebino. Povezave oz. 'varovalke' označite s piko (•). Slika ROM je na hrbtni strani.

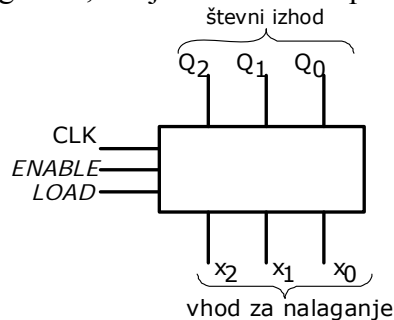
2. Narišite vezje 2-bitnega SIPO (ang. Serial In - Parallel Out) pomikalnega registra z JK-FF in logičnimi vrati. Z izdelanim registrom realizirajte sekvenčno vezje z vhodom  $x$  in izhodom  $y$ , ki postavi  $y=1$ , kadar sta dva zaporedna vhoda enaka in različna od enega prej, torej  $x(t) = x(t-1) \neq x(t-2)$ . Sicer je  $y(t) = 0$ . V začetnem stanju pri  $t=0$  predpostavite, da je vsebina pomikalnega registra enaka '0'.



3. Prikažite sintezo 3-bitnega sinhronega števca navzgor z omogočanjem štetja (*ENABLE*) in vzporednim nalaganjem (*LOAD*) z D flip-flopi, izbiralniki 2/1 in logičnimi vrati. Logika vseh krmilnih signalov je pozitivna.

S tovrstnim števcem realizirajte števec, ki šteje 2, 3, 4, 5, 2, 3, 4, 5 ...

Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



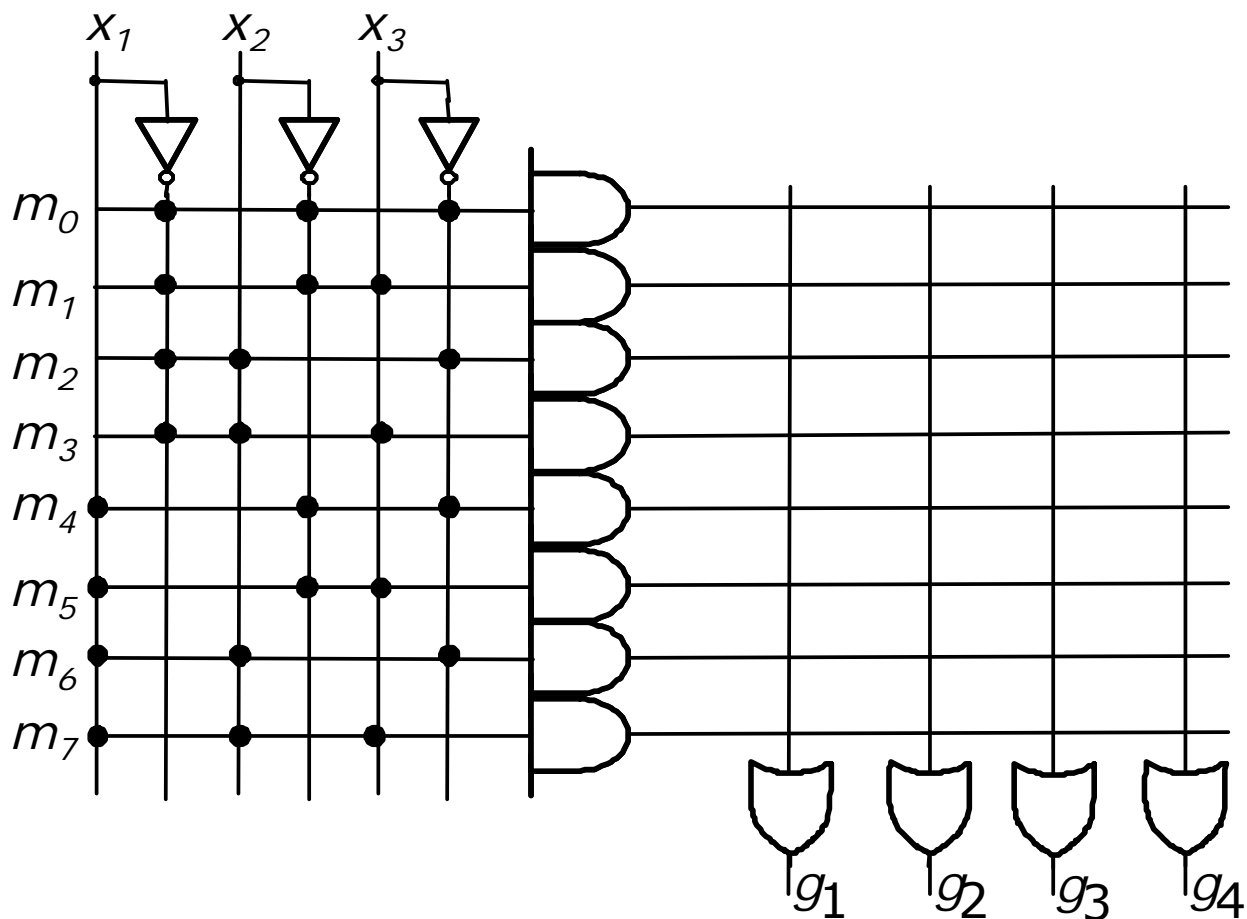
4. Načrtajte tabelo prehajanja stanj Moore-ovega avtomata, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov.

Krmilje ima:

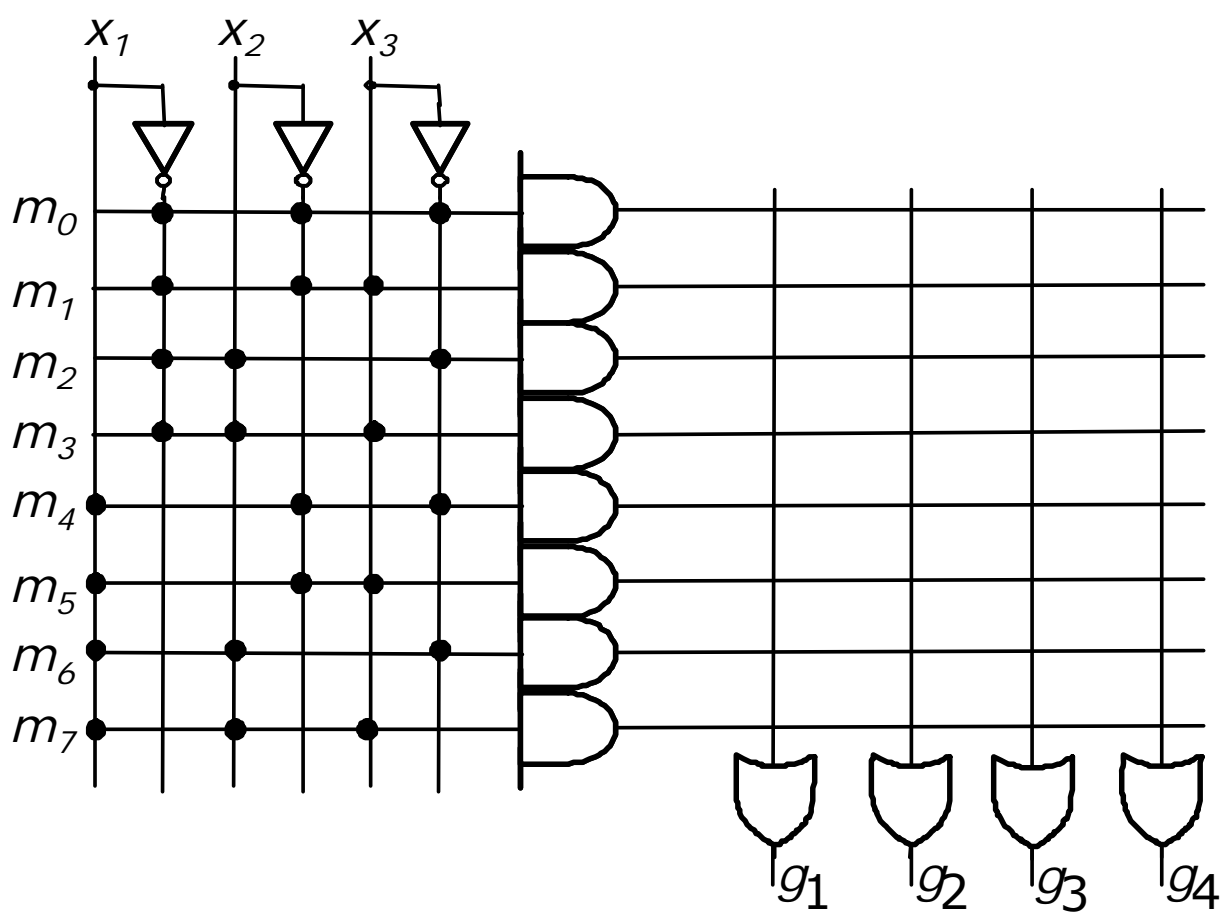
- vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov,
- vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov,
- izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

Avtomat ne vrača drobiža in se ob detekciji plačila 15 centov ne vrača nazaj v začetno stanje, ampak ostane v končnem stanju. Vnos dveh kovancev naenkrat ni mogoč.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si/>



**Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!**



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si/>

# DEVELOPMENT OF DIGITAL SYSTEMS

Midterm Examination  
16. 1. 2014

1. Implement following functions using an imaginary ROM circuit:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

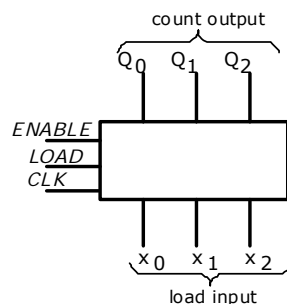
Imaginary ROM circuit has 3 inputs and 4 bit contents as shown on the exam back page. Program the functions by marking programming points with (●) symbol in the OR matrix.

2. Draw the circuit diagram of a two bit SIPO (serial in – parallel out) shift register using JK type flip-flops and logic gates. The register has a serial input ( $SI$ ) and parallel output ( $Q_0, Q_1, Q_2$ ).

Design a sequential circuit with an input  $x$  and output  $y$  using this shift register. The output will become  $y='1'$  when two consecutive inputs  $x(t), x(t-1)$  are equal, and different from previous input  $x(t-2)$ :  $x(t) = x(t-1) \neq x(t-2)$ . Otherwise  $y$  equals '0'. Assume that initially the shift register is empty (its entire contents reads '0').



5. Synthesize a 3 bit binary up counter with count enable signal (ENABLE) and parallel load capability (LOAD) using D type flip-flops, multiplexers 2/1 and logic gates: Draw the state transition table, determine D flip-flop input equations and draw the resulting counter circuit using signal names in the figure below. All control signals have positive logic. Using this counter, design a counter which will count in repeating sequence 2, 3, 4, 5, 2, 3, 4, 5

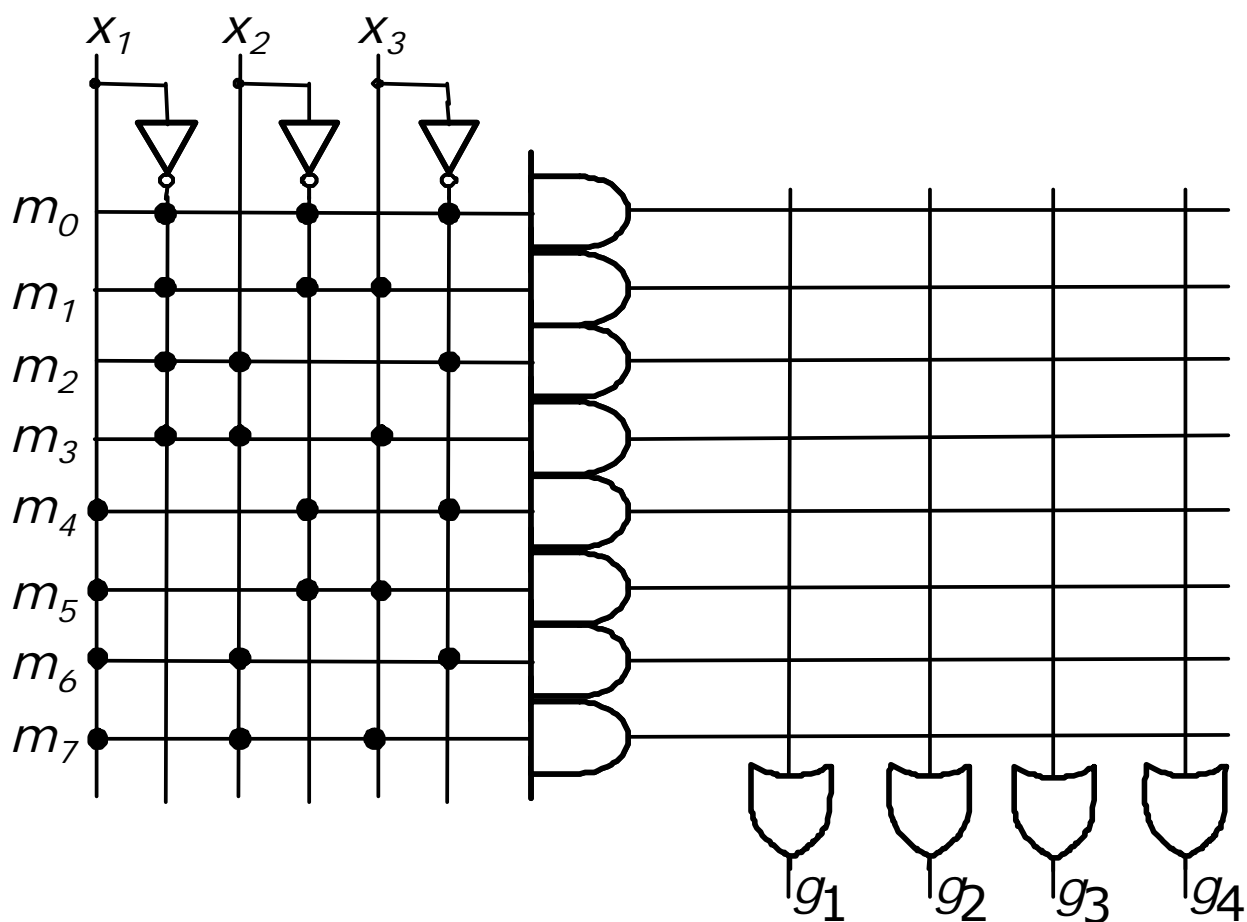


3. Draw a state transition table of a Moore type finite state machine, which controls the operation of a coffee vending machine. The price of coffee is 15 cents. We can pay using 5 or 10 cent coins. The vending machine control circuit has:

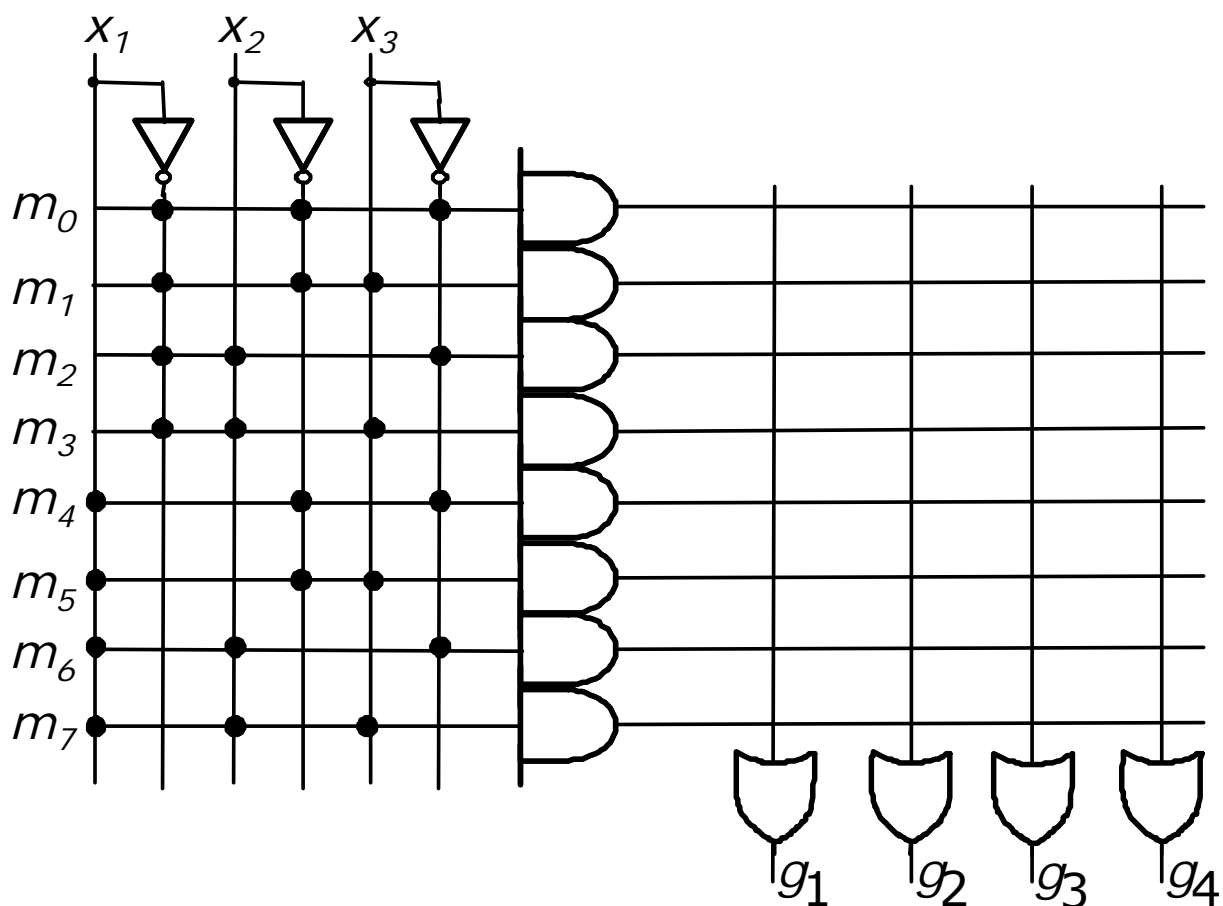
- input *5cent*, which becomes '1', when user inserts a 5 cent coin into machine,
- input *10cent*, which becomes '1', when user inserts a 10 cent coin into machine,
- output *p*, which becomes '1', when user inserts a total sum of 15 cents.

The machine does not return any money and stays in the final state upon payment of 15 cents. Simultaneous insertion of two coins is not possible.

Examination duration is 60 minutes. Each assignment is worth 10 points. Please write your enrollment number and course name on the answer sheet. Grades will be announced on <https://studij.fe.uni-lj.si/>



Upon error, use the other scheme!



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si/>

## Rešitev 1. naloge:

Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ov diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$\begin{aligned}g_1(x_1, x_2, x_3) &= x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= V(4, 6, 5, 7, 0)\end{aligned}$$

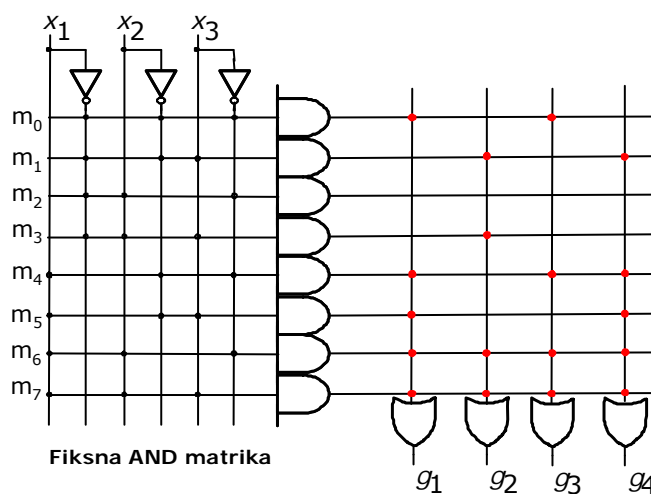
Podobno storimo še za preostale funkcije:

$$\begin{aligned}g_2 &= \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) \\g_2(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\g_2(x_1, x_2, x_3) &= V(1, 3, 6, 7)\end{aligned}$$

$$\begin{aligned}g_3 &= \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 = \\g_3(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) = \\g_3(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\g_3(x_1, x_2, x_3) &= V(0, 4, 6, 7)\end{aligned}$$

$$\begin{aligned}g_4(x_1, x_2, x_3) &= \overline{x_2} \cdot x_3 + x_1 \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 \\g_4(x_1, x_2, x_3) &= V(1, 5, 4, 6, 7)\end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ( $x_1x_2x_3$ ) od  $m_0$  do  $m_7$ . Številka minterma določa naslov lokacije ROM pomnilnika.



Narišemo celotno vezje ROM strukture in vstavimo pike (•) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

Vsebino ROM pomnilnika običajno podajamo s tabelo v kateri programirane povezave (●) v OR matriki pišemo kot '1' na danem mestu vsebine.

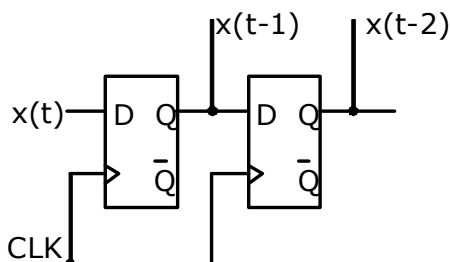
<i>Minterm</i> $g_i(x_1x_2x_3)$	<i>Naslov lokacije</i> $x_1x_2x_3$	<i>Vsebina</i> <i>lokacije</i> $g_1g_2g_3g_4$
$m_0$	$000_2$	$1010_2$
$m_1$	$001_2$	$0101_2$
$m_2$	$010_2$	$0000_2$
$m_3$	$011_2$	$0100_2$
$m_4$	$100_2$	$1011_2$
$m_5$	$101_2$	$1001_2$
$m_6$	$110_2$	$1111_2$
$m_7$	$111_2$	$1111_2$

Realni ROM elementi imajo 8 bitne podatke, zloge ali včasih oktete (ang. *byte*, *octet*). Vsebina ROM elementov se podaja v datoteki, ki jo nato programiramo z posebnim inštrumentom (ROM programatorjem).

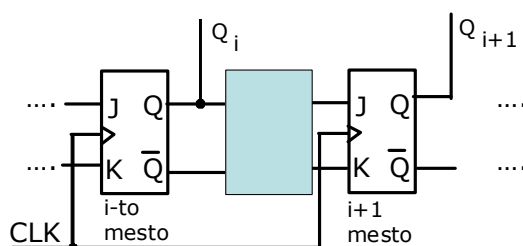
Najenostavnejši način podajanja zapisa vsebine ROM elementa je v surovi dvojiški obliki (ang. *raw binary file*) v kateri si 8 bitni podatki sledijo zapisani v dvojiški obliki. Kompleksnejša zapisa podajanja vsebine ROM s tabelo sta INTEL šestnajstiška oblika (ang. [\*Intel hex record\*](#)) in Motorola SREC oblika (ang. [\*Motorola S record\*](#)).

## Rešitev 2. naloge:

Pomnjenje preteklih vrednosti vhoda  $x$  lahko izvedemo s pomočjo pomikalnega registra.

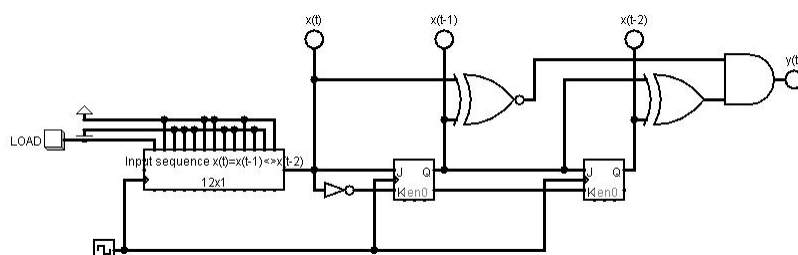


Zapomniti si moramo trenutno vrednost  $x(t)$  ter še dve prejšnji  $x(t-1)$  in  $x(t-2)$ . Za pomnjenje dveh preteklih vrednosti bomo načrtali 2-bitni pomikalni register. Zapišemo tabelo prehajanja stanj z  $i$ -tega mesta na  $(i+1)$  mesto. Enačba izhoda za  $(i+1)$  mesto pomikalnega registra je enačba D-FF ( $Q_{i+1}(t+1) = D_{i+1}$ ), saj je pomikalni register običajno izveden kot veriga D-FF, pri katerih je izhod prejšnjega FF vezan na vhod naslednjega. Ena možnost realizacije je realizacija D-FF z uporabo JK-FF. Ta možnost je potratna, zato raje zapišemo vzbujalno tabelo za opazovano mesto in določimo vhode glede na spremembo stanja ( $Q_{i+1}(t) \rightarrow Q_{i+1}(t+1)$ ) na  $(i+1)$  mestu.



Trenutna vsebina registra		Vsebinska registra ob pomiku	Vhoda FF na mestu $i+1$		Pomen stanja
$Q_i(t)$	$Q_{i+1}(t)$	$Q_{i+1}(t+1)$	$J_{i+1}$	$K_{i+1}$	
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

Če v dobljenih vseh za mesto  $(i+1)$  izberemo primerne redundance, dobimo vhoda  $J_{i+1} = Q_i(t)$  in  $K_{i+1} = Q_i(t)'$ . To velja tudi za vhodno mesto, torej bomo na vhodni JK-FF vezali vhod  $J_0 = x(t)$  in  $K_0 = x(t)'$ . Primerjavo neenakosti mest izvedemo z XOR vrati ( $x(t) \oplus x(t-1)$ ), medtem ko primerjavo enakosti izvedemo z XNOR vrati ( $x(t-1) \oplus x(t-2)$ ). Veljati morata oba pogoja, kar izvedemo z AND vrati.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

Vezje generatorja je v predlogah avditornih vaj na domači strani predmeta:  
Logisim\shift\_reg\sequence\_detector\_x(t)\_is\_x(t\_1)\_is\_not\_x(t\_2).circ

Seveda poleg take rešitve vedno obstaja klasični pristop reševanja s teorijo avtomatov, vendar smo pri tem precej omejeni. Čim bi morali primerjati dlje v zgodovino npr.  $x(t-4)$ ,  $x(t-5)$  ... klasična teorija avtomatov hitro pripelje do minimizacije v Veitch–evih diagramih 5, 6 ... spremenljivk, kar je dolgotrajen proces.

Vhod in trenutno stanje			Naslednje stanje (ob pomiku)		Vhodi JK–FF			
$x$	$Q_1(t)$	$Q_2(t)$	$Q_1(t+1)$	$Q_2(t+1)$	$J_1$	$K_1$	$J_2$	$K_2$
0	0	0	0	0	0	X	0	X
0	0	1	0	0	0	X	X	1
0	1	0	0	1	X	1	1	X
0	1	1	0	1	X	1	X	0
1	0	0	1	0	1	X	0	X
1	0	1	1	0	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	1	1	X	0	X	0

Če primerno izberemo redundance, niti ni potrebno risati Veitch–evih diagramov, ampak lahko dobimo enako rešitev kot pri realizaciji s pomočjo pomikalnega registra:

$$J_1 = x \quad K_1 = \bar{x}$$

$$J_2 = Q_1(t) \quad K_2 = \overline{Q_1(t)}$$

Primerjavo na izhodu tako dobljenega registra ( $x$ ,  $Q_1$ ,  $Q_2$ ) v splošnem izvedemo s primerjalnikom enakosti (ang. equality comparator), realiziranega z XOR oz. z XNOR vrati.



### Rešitev 3. naloge:

trenutno stanje			naslednje stanje			D-FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Podobno za D<sub>2</sub> narišemo Veitchev diagram

$D_2$ :

	$Q_2$			
$Q_1$	1	0	1	0
	1	1	0	0
	$Q_0$			

Za D<sub>2</sub> sledi:

$$D_2 = Q_2 Q_1' + Q_2 Q_0' + Q_2' Q_1 Q_0$$

Iz tabele prehajanja stanj števca določimo enačbe D-FF:

Za D<sub>0</sub> se iz tabele vidi  $D_0 = Q_0'$

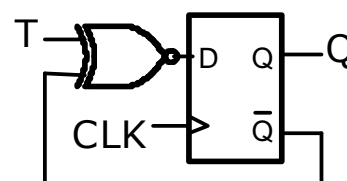
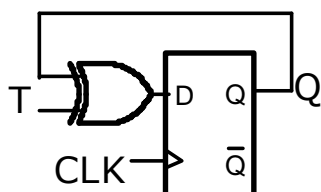
Za D<sub>1</sub> narišemo Veitchev diagram:

$D_1$ :

	$Q_2$			
$Q_1$	1	0	0	1
	0	1	1	0
	$Q_0$			

$$D_1 = Q_0 \oplus Q_1$$

Če enačbo  $D_0 = Q_0'$  zapišemo kot  $D_0 = 1 \oplus Q_0$  in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:

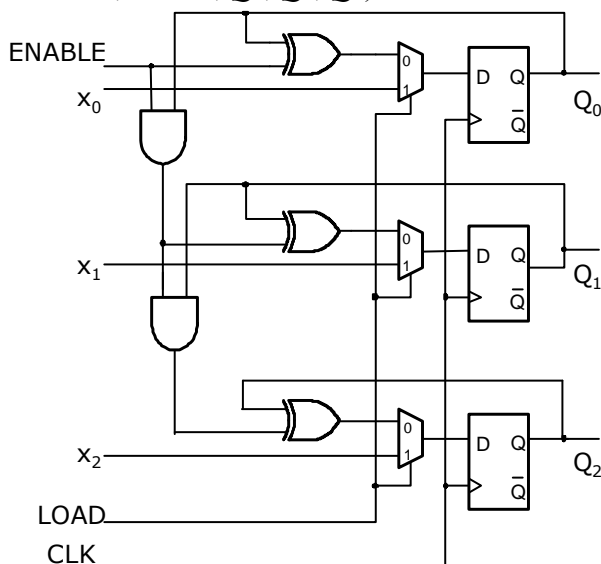


Slika 1: Realizacija T-FF s pomočjo D-FF in XOR vrat (levo) in XNOR vrat (desno)

Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (*ENABLE*): Če prvemu "T-FF" (D-FF z XOR vrati) postavimo vhod  $T_0 = 0'$  namesto  $T_0 = 1'$ , vsi FF ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod  $T_0$  postavimo zunanji signal *ENABLE*, števec ne bo štel, ampak ohranjal stanje, če bo *ENABLE* = '0'. V verigi sinhronnega števca so namreč vsi FF vezani tako, da so odvisni od prvega FF. Če stanje ohranja prvi, ga bodo tudi vsi ostali.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnega vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame za 5 spremenljivk ( $ENABLE$ ,  $LOAD$ ,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



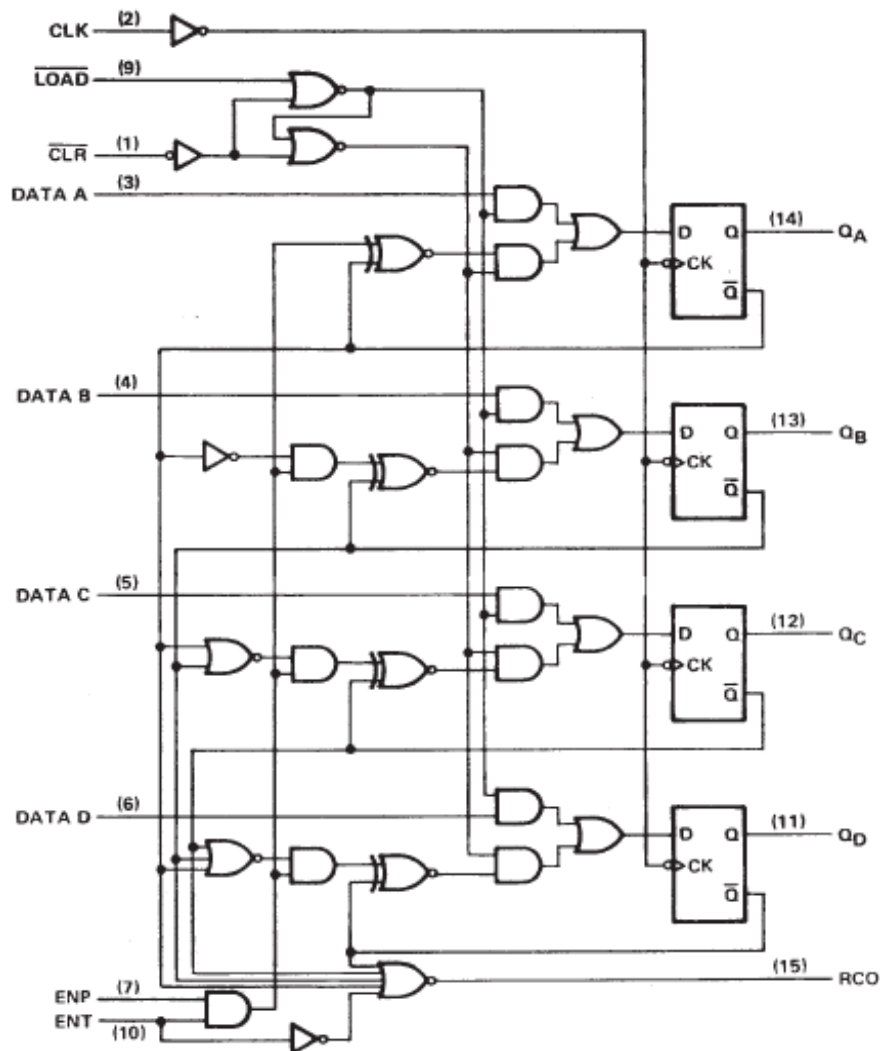
**Slika 2: Sinhroni števec z vzporednim nalaganjem (LOAD) in omogočanjem štetja (ENABLE) (3-bitna izvedba).**

Če želimo z nastalim števcem šteti naraščajoče 2, 3, 4, 5 ..., moramo ko števec pride do stanja 5 ( $Q_2Q_1Q_0=101_2$ ) števec postaviti nazaj na stanje v stanje 2 ( $Q_2Q_1Q_0=x_2x_1x_0=010_2$ ), torej signal  $LOAD$  dekodiramo s pomočjo 2–vhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se  $Q_2 = '1'$  in  $Q_0 = '1'$  v števeni sekvenci pojavlja samo enkrat – če bi se večkrat, bi morali dekodirati tudi  $Q_1$ . Pri tovrstnih števcih ponavadi uporabljamo še en signal  $RESET$ , s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod D–FF za brisanje asinhrono priključimo signal  $RESET$ .

Nastali strukturi števca bi lahko na isti način dodali še četrti bit. Tako bi dobili podobno strukturo kot je 4-bitni sinhroni števec z vzporednim nalaganjem 74163<sup>1</sup>. Pri spodnji realizaciji tega vezja so uporabljeni D–FF, proženi na negativni rob signala ure ( $CLK$ ). Štetje omogočimo s signaloma  $ENP$ ,  $ENT$  (ang. enable parallel, enable transfer). Štetje je izvedeno tako, da se D–FF spremenijo v T–FF, kar dosežemo s pomočjo XNOR vrat, ki imajo en vhod vezan na števeni signal ( $ENT=ENP='1'$ ), drug vhod pa na izhod  $Q'$  FF. AND vrata pred XNOR zagotavljajo prenehanje štetja, če velja  $ENT \cdot ENP \neq '1'$ . Na vhodu D–FF je vezan enostaven izbiralnik iz dveh AND vrat, vezanih na OR vrata. Ta izbiralnik določa, ali bo števec štel, ali bo vzporedno naložil vrednost. Zgornja AND vrata izbiralnika so krmiljena s signalom  $LOAD \cdot CLR'$ , spodnja pa z  $LOAD' \cdot CLR'$ . Čim velja, da je  $CLR' = '1'$  in  $LOAD' = '0'$ , se v D–FF asinhrono naloži vsebina na vseh  $Q_D$   $Q_C$   $Q_B$   $Q_A = DATA_D$   $DATA_C$   $DATA_B$   $DATA_A$ , medtem ko dokler velja  $LOAD' = '1'$  in  $CLR' = '1'$ , bo števec štel navzgor. Če je  $CLR' = '0'$ , je na obeh vseh AND vrat izbiralnika '0' in stanje vseh D–FF se asinhrono postavi na  $Q_DQ_CQ_BQ_A = "0000"$ . Števni izhodi so  $Q_D$   $Q_C$   $Q_B$   $Q_A$ .

<sup>1</sup> <http://www.alldatasheet.com/view.jsp?Searchword=74163>

Izhod *RCO* (*ripple carry out*) se postavi na '1', ko je števec v stanju "1111" ob tem, da je  $ENT='1'$ . Signal *RCO* je izveden z NOR vrati na vhode katerih so priključeni negirani izhodi D-FF, kar je po De Morgan-ovem teoremu ekvivalent AND vratom.

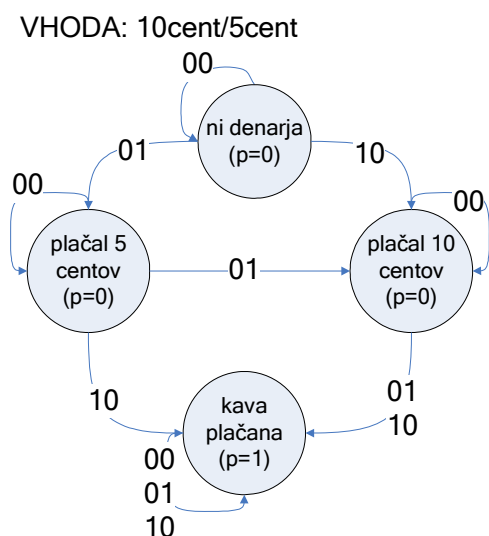


Slika 3: Struktura 4-bitnega MSI sinhronnega števca z vzporednim nalaganjem (74163).

#### Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj. Opis diagrama stanj:

Na začetku se nahajamo v stanju "ni denarja", v katerem je izhod  $p=0$ . Vhoda v avtomat sta dva: 10cent in 5cent, kar na diagramu kodiramo kot 10cent/5cent.



Mehanizem za vnos kovancev preprečuje hkraten vnos dveh kovancev, torej je kombinacija (10cent/5cent=11) nemogoča, zato bo avtomat od tu lahko prešel v poljubno stanje (X). Če uporabnik ni vrgel denarja v avtomat (10cent/5cent=00), potem ostaja v stanju "ni denarja". Če uporabnik vrže v avtomat 5 centov (10cent/5cent=01), potem preide v stanje "plačal 5 centov". Če uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "plačal 10 centov".

Ne glede na to koliko je vrgel bo izhod v teh dveh stanjih enak  $p=0$ , ker še ni plačal celotne cene kave. Če smo v stanju "plačal 5 centov" in uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ). Stanje "kava plačana" je končno in tam tudi ostanemo za vse možne kombinacije. Če smo v stanju "plačal 10 centov" in uporabnik vrže v avtomat 5 ali 10 centov (10cent/5cent=10 oz. 01), potem podobno preidemo v stanje "kava plačana", kjer postavimo izhod ( $p=1$ ).

Naredimo tabelo prehajanja stanj:

trenutno stanje	10cent	5cent	naslednje stanje	izhod p
ni denarja	0	0	ni denarja	0
ni denarja	0	1	plačal 5 centov	0
ni denarja	1	0	plačal 10 centov	0
ni denarja	1	1	X	X
plačal 5 centov	0	0	plačal 5 centov	0
plačal 5 centov	0	1	plačal 10 centov	0
plačal 5 centov	1	0	kava plačana	0
plačal 5 centov	1	1	X	X
plačal 10 centov	0	0	plačal 10 centov	0
plačal 10 centov	0	1	kava plačana	0
plačal 10 centov	1	0	kava plačana	0
plačal 10 centov	1	1	X	X
kava plačana	0	0	kava plačana	1
kava plačana	0	1	kava plačana	1
kava plačana	1	0	kava plačana	1
kava plačana	1	1	X	X

Izberemo kodiranje stanj:

stanje	$Q_1$	$Q_0$
ni denarja	0	0
plačal 5 centov	0	1
plačal 10 centov	1	0
kava plačana	1	1

Nad tabelo prehajanja stanj uporabimo predlagano kodiranje stanj:

$Q_1$	$Q_0$	10cent	5cent	$Q_1$	$Q_0$	izhod P
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	X	X	X
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	X	X	X

Naloga zahteva realizacijo z D–FF:

t				t+1				
$Q_1$	$Q_0$	10cent	5cent	$Q_1$	$Q_0$	$D_1$	$D_0$	izhod P
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	1	X	X	X	X	X
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	1	1	0
0	1	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0	0
1	0	0	1	1	1	1	1	0
1	0	1	0	1	1	1	1	0
1	0	1	1	X	X	X	X	X
1	1	0	0	1	1	1	1	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na <http://estudent.fri.uni-lj.si/fe.html>

1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	X	X	X	X	X

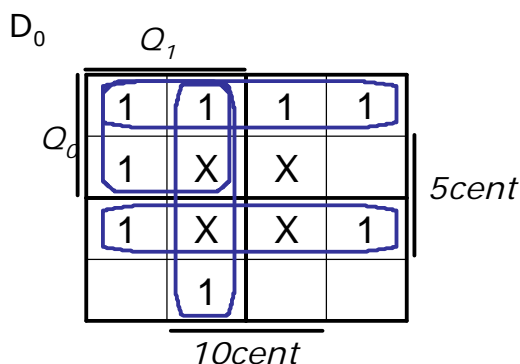
Iz dobljene tabele narišemo Veitch–eve diagrame za oba D–FF in izhod p:

$$D_0 = V(1, 4, 6, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$D_1 = V(2, 5, 6, 8, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$p = V(12-14) \text{ in } Vx(3, 7, 11, 15)$$

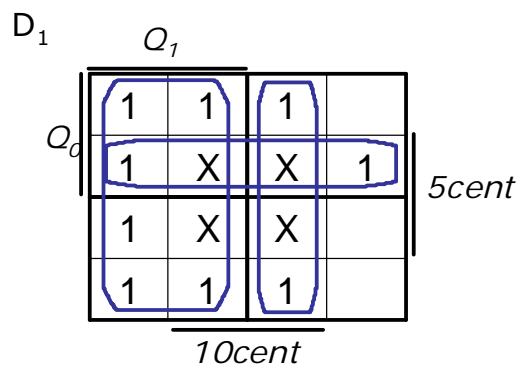
Veitch–ev diagram za  $D_0$ :



$$D_0 = Q_1 \cdot Q_0 + \overline{5cent} \cdot Q_0 + 10cent \cdot Q_1 + 5cent \cdot \overline{Q_0}$$

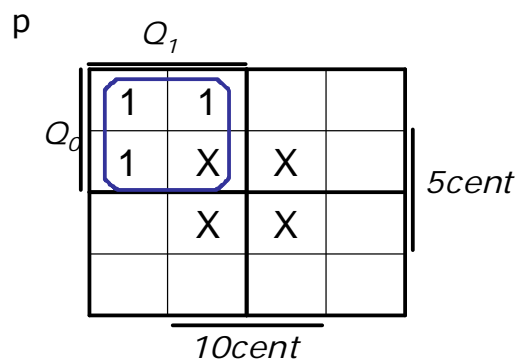
$$D_0 = Q_1 \cdot (Q_0 + 10cent) + 5cent \oplus Q_0$$

Veitch–ev diagram za  $D_1$ :



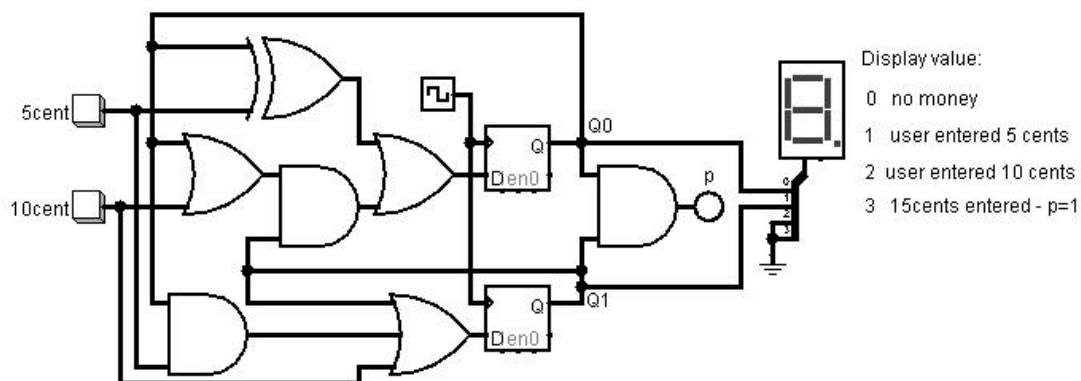
$$D_1 = Q_1 + 5cent \cdot Q_0 + 10cent$$

Veitch–ev diagram za izhod p:



$$p = Q_1 \cdot Q_0$$

Enačbo za izhod  $p$  bi lahko napisali tudi samo s sklepanjem, saj se izhod  $p$  postavi samo, ko je avtomat v stanju "kava plačana", ki ima kodo  $Q_1Q_0="11"$  – torej ko bosta  $Q_1$  in  $Q_0$  enaka '1', bo izhod  $p=1$ .



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta: `Logisim\fsm\Vending_machine_d_ff_5_10_cents_price_15cents.circ`

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
18. 12. 2014

1. Določite popolno konjunktivno normalno obliko (PKNO) in popolno disjunktivno normalno obliko (PDNO) funkcije  $f$  z uporabo pravil Boole-ove logike.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

2. Realizirajte podano funkcijo  $f$  z redundancami z eno 4-bitno aritmetično-logično enoto (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \quad \text{in} \quad V_x(3, 15)$$

3. Z uporabo "carry lookahead" načina seštevanja seštejte števili  $x$  in  $y$ . Na vseh mestih določite vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ). Končni rezultat izračunajte z uporabo prenosov ( $c_i$ ), ki izvirajo iz pomočjo funkcij ( $g_i$ ) in ( $p_i$ ), ter vhodnih operandov  $x$  in  $y$ .

$$x + y = 23_{10} + 45_{10} = 68_{10}$$

4. Realizirajte funkcijo  $f$  z enim izbiralnikom 4/1.

$$f(a, b, c, d) = (a \cdot \overline{b} + b \cdot c \cdot \overline{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\overline{c} + d))$$



# DEVELOPMENT OF DIGITAL SYSTEMS

Midterm Examination

18. 12. 2014

1. Determine the minimal normal form (MNO) of a given incompletely specified function  $f$ . Write the *complete* POS (product-of-sums) and *complete* SOP (sum-of-products) form of a given function  $f$  using Boolean algebra rules.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

2. Implement the incompletely specified function  $f$  using a single four bit arithmetic logic unit (ALU). All possible variable negations have to be implemented using this ALU.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Add two unsigned numbers  $x$  and  $y$  using carry look-ahead addition technique. Evaluate generate ( $g_i$ ) and propagate ( $p_i$ ) functions for all bit positions. Calculate the sum ( $S_i$ ) using carry bits ( $c_i$ ) and both inputs ( $x_i, y_i$ ). Carry bits ( $c_i$ ) have to be evaluated using generate ( $g_i$ ) and propagate ( $p_i$ ) functions.

$$x + y = 23_{10} + 45_{10} = 68_{10}$$

4. Implement the function  $f$  using a single 4/1 multiplexer.

$$f(a, b, c, d) = (a \cdot \overline{b} + b \cdot c \cdot \overline{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\overline{c} + d))$$

## Rešitev 1. naloge

Funkcija je zapisana v večnivojski obliki, torej jo izrazimo v normalno obliko.

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_2) \cdot \overline{x_3} + ((\overline{x_2} \equiv x_4) \downarrow \overline{x_1})$$

Funkciji NOR ( $\downarrow$ ) in ekvivalence ( $\equiv$ ) izpišemo:

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 + x_2}) \cdot \overline{x_3} + \left( \overline{(\overline{x_2 \oplus x_4}) + x_1} \right)$$

Ekvivalenco smo izrazili kot negacijo XOR. Uporabimo De Morganov teorem:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2 \oplus x_4}) \cdot x_1$$

Izpišemo enačbo funkcije XOR ( $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$ ) in dobimo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + (\overline{x_2} \cdot \overline{x_4} + x_2 \cdot x_4) \cdot x_1$$

Razširimo še zadnjo konjunkcijo in rezultat je oblika MDNO:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_4} \cdot x_1 + x_1 \cdot x_2 \cdot x_4$$

Če uporabimo lastnost Boole-ove algebre ( $\overline{\overline{a}} + a = 1$ ) lahko zapišemo:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4} + x_1 \cdot x_2 \cdot x_3 \cdot x_4 + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4$$

Kar lahko zapišemo v obliki PDNO:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} + x_1 \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4} + x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_2 \cdot x_3 \cdot x_4 \\ f_{PDNO}(x_1, x_2, x_3, x_4) &= V(0, 1, 8, 10, 13, 15) \end{aligned}$$

PDNO pretvorimo v PKNO tako, da pregledamo manjkajoče minterme: 2, 3, 4, 5, 6, 7, 9, 11, 12, 14. Te minterme preslikamo preko tabele:

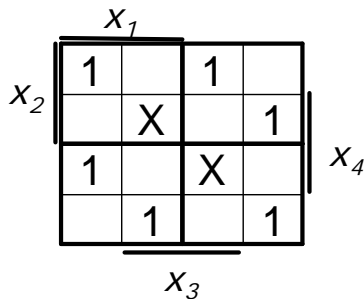
$m_i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$M_i$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Funkcija v PKNO se torej glasi:

$$\begin{aligned} f_{PDNO}(x_1, x_2, x_3, x_4) &= V(0, 1, 8, 10, 13, 15) \\ f_{PKNO}(x_1, x_2, x_3, x_4) &= \&(13, 12, 11, 10, 9, 8, 6, 4, 3, 1) \end{aligned}$$

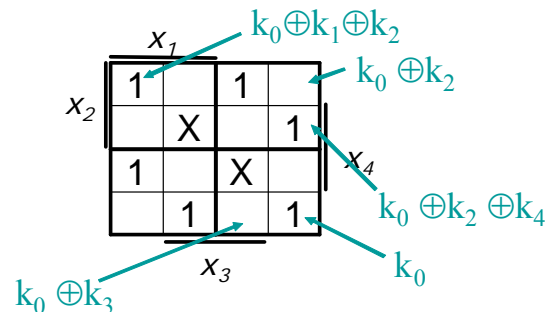
## Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitch–ev diagram:



Funkcija vsebuje same diagonalne člene, zato realizacija v obliki KNO oz. DNO ne nudi minimalne oblike. Če se izkaže, da je funkcija linearna, jo lahko realiziramo s pomočjo XOR funkcij. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

S pomočjo Veitch–evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

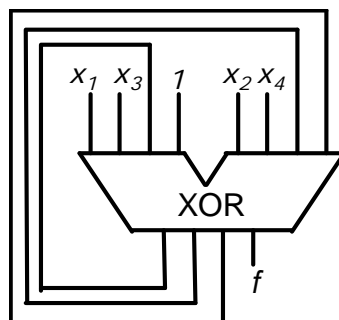
Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Pri realizaciji uporabimo lastnost združevanja, ki velja za XOR funkcijo.

$$f(x_1, x_2, x_3, x_4) = 1 \oplus ((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$$



### Rešitev 3. naloge:

Uporaba "carry look ahead" načina seštevanja zahteva določitev vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ). Končni rezultat izračunamo z uporabo prenosov, ki izvirajo iz pomožnih funkcij ( $g_i$ ) in ( $p_i$ ), ter vhodnih operandov  $x$  in  $y$ . Bistvena prednost tega načina seštevanja je, da prenose lahko izračunamo vnaprej (ang. look ahead) vzporedno na osnovi enačb, s katerimi dani prenos izrazimo kot funkcijo ( $g_i$ ) in ( $p_i$ ) ter vhodnega prenosa. Zaradi vzporednega načina izračuna prenosov je izračun vsote neodvisen od števila mest seštevalnika. Problem tega načina so kompleksne funkcije, ki nastanejo pri naraščajočem številu mest seštevanja, zato se CLA seštevalniki združujejo po 4 mesta, izhodi za funkcije tvorjenja ( $g$ ) in širjenja ( $p$ ) teh CLA pa se vežejo na generator izhodnih prenosov (ang. carry look ahead generator oz. CLAG). Način seštevanja "carry look ahead" zahteva določitev vrednosti funkcij tvorjenja ( $g_i$ ) in širjenja ( $p_i$ ) ter prenosa na naslednje mesto ( $c_{i+1}$ ) za vsako mesto posebej.

Osnovne enačbe "carry look ahead" so:

$$g_i = x_i \cdot y_i$$

$$p_i = x_i \oplus y_i$$

$$c_{i+1} = g_i + p_i \cdot c_i$$

Pri seštevanju upoštevamo, da vhodnega prenosa ni ( $C_{in}=0$ ).

Preostali prenosi se izračunajo vzporedno iz izrazov iterativnega vstavljanja formule za prenos  $c_{i+1}$  od mesta 0 do mesta 7 (mesto izhodnega prenosa). Slednje je bistvena prednost "carry look ahead" načina seštevanja pred običajnim "ripple carry" načinom seštevanja.

$$c_0 = c_{in} = 0$$

$$c_1 = g_0 + p_0 \cdot c_0 = 1 + 0 \cdot 0 = 1$$

$$c_2 = g_1 + p_1 \cdot c_1 = 0 + 1 \cdot 1 = 1$$

$$c_3 = g_2 + p_2 \cdot c_2 = 1 + 0 \cdot 1 = 1$$

$$c_4 = g_3 + p_3 \cdot c_3 = 0 + 1 \cdot 1 = 1$$

$$c_5 = g_4 + p_4 \cdot c_4 = 0 + 1 \cdot 1 = 1$$

$$c_6 = g_5 + p_5 \cdot c_5 = 0 + 1 \cdot 1 = 1$$

$$c_7 = g_6 + p_6 \cdot c_6 = 0 + 0 \cdot 1 = 0$$

$i$	7	6	5	4	3	2	1	0
$x_i$		0	0	1	0	1	1	1
+		0	1	0	1	1	0	1
$g_i$		0	0	0	0	1	0	1
$p_i$		0	1	1	1	0	1	0
$c_i$	0	1	1	1	1	1	1	0
$S_i$	0	1	0	0	0	1	0	0

Mesta vsote  $S_i$  dobimo z XOR operacijo  $S_i = x_i \oplus y_i \oplus c_i$  pri čemer XOR predstavlja vsoto po modulu 2 oz. funkcijo v PDNO:  $S_i(x_i, y_i, c_i) = V(1, 2, 4, 7)$ .

Rešitev 4. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot ((a \cdot c \cdot d) \cdot (\bar{c} + d))$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo desni člen funkcije in uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$ , lastnost  $x \cdot x = x$  in lastnost  $0 + x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d \cdot \bar{c} + a \cdot c \cdot d \cdot d)$$

Nad rezultatom ponovno uporabimo lastnost Boole-ove logike  $x \cdot \bar{x} = 0$  in lastnost  $x \cdot x = x$ .

$$f(a,b,c,d) = (a \cdot \bar{b} + b \cdot c \cdot \bar{d} + b \cdot c) \cdot (a \cdot c \cdot d)$$

Rezultat vnesemo v levi del funkcije in znova uporabimo omenjene lastnosti Boole-ove logike:

$$f(a,b,c,d) = (a \cdot \bar{b} \cdot a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot a \cdot c \cdot d + b \cdot c \cdot a \cdot c \cdot d)$$

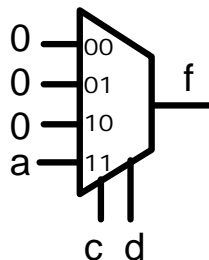
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre  $x + \bar{x} = 1$ :

$$f(a,b,c,d) = a \cdot \bar{b} \cdot c \cdot d + b \cdot a \cdot c \cdot d$$

$$f_{PDNO}(a,b,c,d) = V(11, 15)$$

$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

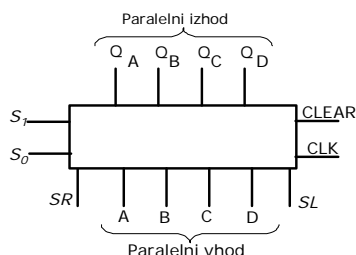
in jo realiziramo z enim izbiralnikom 4/1, tako da naredimo Shannon-ov razvoj funkcije. Glede na kombinacijo naslovnih vhodov izbiralnika dobimo 6 možnih rešitev (ac, ca, ad, da, cd, dc).



# RAZVOJ DIGITALNIH SISTEMOV

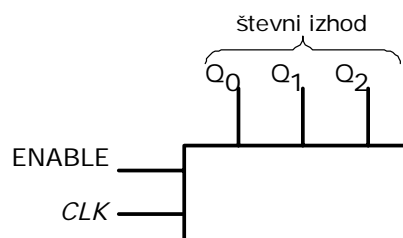
2. kolokvij  
23. 1. 2015

1. Realizirajte JK-flip flop z uporabo RS- flip flopa in logičnih vrat.
2. Z uporabo D flip-flopov in izbiralnikov 4/1 prikažite sintezo univerzalnega 4-bitnega pomikalnega registra, ki ima dva funkcijska vhoda  $S_0$  in  $S_1$  in opravlja funkcije po spodnji tabeli. Register ima tudi zaporedna vhoda za pomik v levo (SL – serial left), pomik v desno (SR – serial right) in asinhroni vhod za brisanje CLEAR (aktiven nizek).



$S_1$	$S_0$	funkcija
0	0	drži stanje
0	1	pomik vsebine eno mesto desno
1	0	pomik vsebine eno mesto levo
1	1	nalaga vsebino z vhodov ABCD

3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor z omogočanjem štetja (ENABLE) s T flip-flopi. Zapišite tabelo prehajanja stanj, določite enačbe vhodov T-FF in narišite vezje z uporabo signalov na sliki.



4. Narišite **diagram prehajanja stanj** za avtomat končnih stanj, ki ima vhod  $w$  in izhod  $z$ . Avtomat končnih stanj postavi izhod  $z='1'$  takoj, ko se na vhodu pojavi zaporedje **110** ali **101**, sicer je  $z='0'$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda. Delovanje avtomata je povzema podano časovno zaporedje, pri čemer  $t_0 \dots t_{12}$  označujejo aktivne prehode signala ure  $CLK$ :

$CLK$	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
$w$	0	1	1	0	1	1	0	1	1	1	0	0	0
$z$	–	0	0	1	1	0	1	1	0	0	1	0	0

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

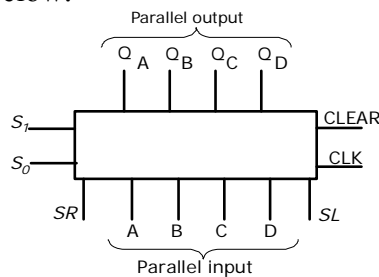
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# DEVELOPMENT OF DIGITAL SYSTEMS

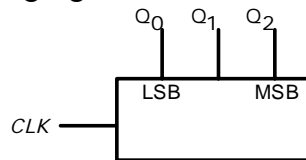
2<sup>nd</sup> Midterm Examination  
23. 1. 2015

1. Implement a JK type flip flop using a single RS flip flop and logic gates.
2. Draw the circuit of a 4 bit universal shift register using D type flip-flops and 4/1 multiplexers. The register comprises four content outputs  $Q_A$ ,  $Q_B$ ,  $Q_C$ ,  $Q_D$  and two function inputs  $S_0$  and  $S_1$ . It also has two shift operation inputs:  $SL$  (shift left) and  $SR$  (shift right). A LOAD operation will load the inputs  $A$ ,  $B$ ,  $C$ ,  $D$  into register locations  $Q_A$ ,  $Q_B$ ,  $Q_C$ ,  $Q_D$  respectively. An active low asynchronous  $CLEAR$  input sets the contents of register to zero. Register operations are specified in a table below.



$S_1$	$S_0$	<i>function</i>
0	0	HOLD
0	1	SHIFT RIGHT
1	0	SHIFT LEFT
1	1	LOAD

3. Synthesize a 3 bit binary up counter with ENABLE using T type flip-flops: Draw the state transition table, determine T flip-flop input equations and draw the resulting counter circuit using signal names in the figure below.



4. Draw a **state transition diagram** of a sequence detector FSM. Sequence detector has an input  $w$  and an output  $z$ . The FSM sets the output  $z='1'$  *immediately* after a sequence **110** or **101** is detected at its input. The sequences may overlap (i.e. input sequence 1101 must be detected twice). FSM operation is described in a test sequence given below, where  $t_0 \dots t_{12}$  denote active edge transitions of clock signal  $CLK$ :

$CLK$	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$	$t_{11}$	$t_{12}$
$w$	0	1	1	0	1	1	0	1	1	1	0	0	0
$z$	—	0	0	1	1	0	1	1	0	0	1	0	0

Examination duration is 60 minutes. Each assignment is worth 10 points.

Please sign your answer sheet using your enrollment number. Solutions will be published on the course web page. Examination results will be announced on the course web page.

## Rešitev 1. naloge

Za vezje JK-FF narišemo pravilnostno tabelo, pri čemer na vhodni strani zberemo vhode J, K in trenutno stanje  $Q(t)$ , na izhodni pa naslednje stanje  $Q(t+1)$ . JK-FF opravlja štiri funkcije (HOLD, RESET, SET, INVERT) glede na kombinacijo vhodnih signalov, medtem ko RS-FF opravlja samo tri (HOLD, RESET, SET). Posamezno kombinacijo prehodov iz  $Q(t)$  v  $Q(t+1)$  je možno doseči na več načinov. Primer je prva kombinacija  $J=K=Q(t)=0$  pri čemer je  $Q(t+1)=0$ . Če je bil izhod FF prej 0 in je tudi v naslednjem stanju 0 sta vzroka za to lahko dva: HOLD ( $R=0, S=0$ ) ali RESET ( $R=1, S=0$ ), kar opišemo s kombinacijo ( $R=X, S=0$ ). Podobno lahko sklepamo za ostale kombinacije vhodov.

J	K	$Q(t)$	$Q(t+1)$	R	S	funkcija RS	funkcija JK
0	0	0	0	X	0	HOLD/RESET	HOLD
0	0	1	1	0	X	HOLD/SET	HOLD
0	1	0	0	X	0	HOLD/RESET	RESET
0	1	1	0	1	0	RESET	RESET
1	0	0	1	0	1	SET	SET
1	0	1	1	0	X	HOLD/SET	SET
1	1	0	1	0	1	SET	INVERT
1	1	1	0	1	0	RESET	INVERT

Iz tabele narišemo Veitcheva diagrama za vhoda R in S v odvisnosti od vhodov J, K in trenutnega stanja  $Q(t)$ .

R:

	J			
K	0	1	1	X
	0	0	0	X
	$Q(t)$			

S:

	J			
K	1	0	0	0
	1	X	X	0
	$Q(t)$			

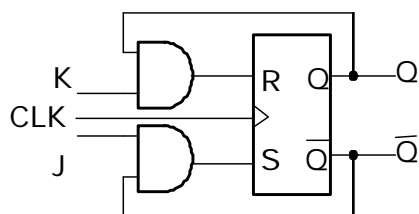
Dobljena diagrama minimiziramo in zapišemo enačbi vhodov J in K:

$$R = K \cdot Q(t)$$

ter

$$S = J \cdot \overline{Q(t)}$$

Vezje narišemo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>



## Rešitev 2. naloge:

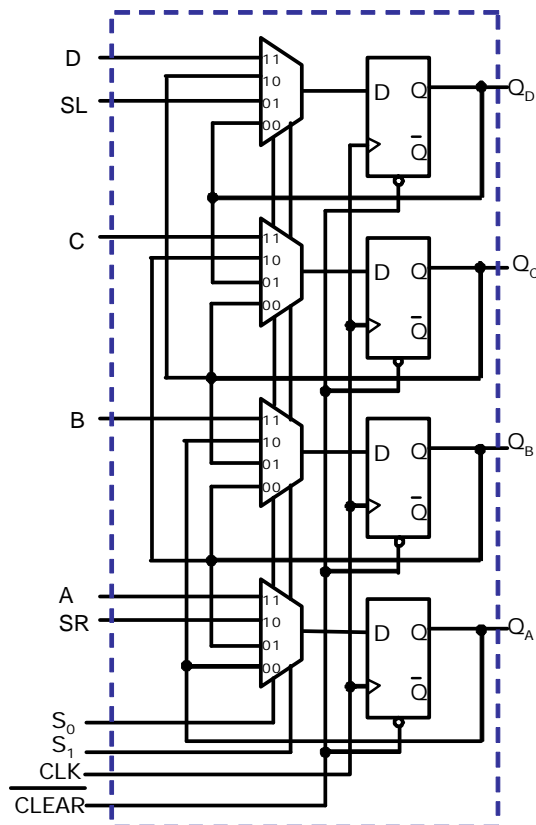
Vsako od operacij izpišemo v pravilnostno tabelo v kateri združimo funkcijska bita  $S_1$  in  $S_0$  in trenutno stanje na  $i$ -tem mestu registra  $Q_i(t)$ . Mesta registra od leve proti desni so  $Q_i = (Q_A, Q_B, Q_C, Q_D)$ . Realizacija z D flip-flopi nam analizo močno poenostavi, zaradi enačbe D flip-flopa:  $D = Q(t+1)$ .

Tabela 1: Prehajanje stanj univerzalnega registra.

$S_1$	$S_0$	$Q_i(t+1)$	funkcija
0	0	$Q_i(t)$	HOLD
0	1	$Q_{i+1}(t)$	LSR
1	0	$Q_{i-1}(t)$	LSL
1	1	$x_i$	LOAD

Iz poenostavljene tabele prehajanja stanj univerzalnega registra sestavimo realizacijo, ki bo vključevala izbiralnike MUX 4/1 in D-FF.

Na naslovna vhoda vseh MUX 4/1 vodimo funkcijska signala  $S_1$  in  $S_0$ . Potem na vsakem podatkovnem vhodu realiziramo ustrezno funkcijo.



Stanje  $S_1S_0 = "00"$  pomeni držanje stanja (*HOLD*), torej bodo trenutne vrednosti D-FF ohranile vrednost  $Q_i(t+1) = Q_i(t)$ . Na sliki to realiziramo tako, da vodimo izhod D-FF nazaj na vhod pri podatkovnem vhodu 00. Stanje  $S_1S_0 = "01"$  pomeni pomik desno (*LSR* – ang. logic shift right), torej bodo D-FF pomaknili vsebino eno mesto desno. Pomik desno pomeni, da na mesto skrajno levega bita vpišemo vrednost zaporednega vhoda SR, nato  $Q_A$  vodimo na vhod  $Q_B$  in tako do skrajno desnega bita. Stanje  $S_1S_0 = "10"$  pomeni pomik levo (*SHL* – ang. shift left), torej bodo D-FF pomaknili vsebino eno mesto levo. Pomik levo pomeni, da na mesto skrajno desnega bita vpišemo vrednost zaporednega vhoda SL, nato  $Q_D$  vodimo na vhod  $Q_C$  in tako do skrajno levega bita.  $S_1S_0 = "11"$  pomeni vzporedno nalaganje z vhodov (*LOAD*)  $Q_D(t+1) = D$ ,  $Q_C(t+1) = C$ ,  $Q_B(t+1) = B$ ,  $Q_A(t+1) = A$ . Na tabeli smo  $i$ -ti vhod za vzporedno nalaganje označili kot  $x_i = (A, B, C, D)$ .

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Normalna analiza bi zahtevala, da narišemo Veitch-eve diagrame za tri spremenljivke za vsak vhod T-FF, vendar ker so T-FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Iz stolpca T<sub>0</sub> se vidi:

$$T_0 = 1$$

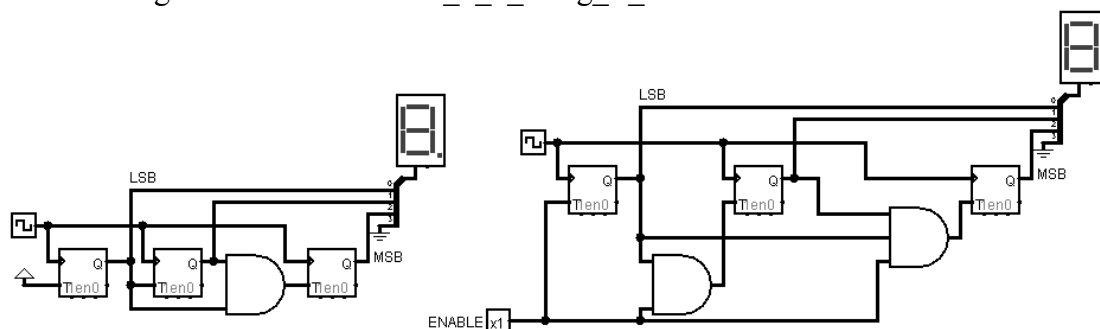
Z opazovanjem stolpcev trenutnega stanja določimo T<sub>1</sub>:

$$T_1 = Q_0$$

Podobno lahko določimo T<sub>2</sub>:

$$T_2 = Q_0 \cdot Q_1$$

Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_7\_0\_using\_T\_FF.circ:



Funkcijo omogočanja štetja bi lahko realizirali s pisanjem tabele prehajanje stanj števca za štiri spremenljivke (ENABLE, Q<sub>0</sub>, Q<sub>1</sub>, Q<sub>2</sub>). Z malo razmišljanja se izognemo dolgotrajnemu pisanju: Števec bo obstal (ENABLE='0'), če so vsi vhodi T-FF enaki '0'. Števec bo štel (ENABLE='1'), ko bodo vhodi T-FF lahko spreminjali stanje, torej se bo izhod prejšnje stopnje nespremenjen pojavil na vhodu naslednje stopnje T<sub>i+1</sub>. Povedano strnemo v tabelo za vhod vsakega T-FF:

ENABLE	Izhod prejšnje stopnje	T <sub>i+1</sub>
0	0	0
0	1	0
1	0	0
1	1	1

Funkcijo ENABLE realiziramo z dvovhodnimi AND vrati pred vhodom posameznega T-FF. Na vhoda AND vrat vodimo ENABLE in izhod iz prejšnje stopnje štetja, kot kaže desni del slike. Na LSB mestu štetja AND vrat ne potrebujemo, ampak ENABLE priključimo neposredno na vhod T-FF.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

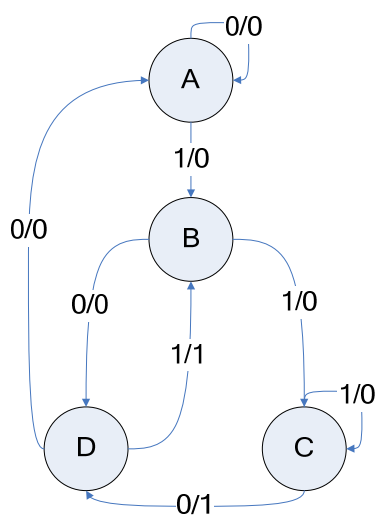
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Naloga zahteva realizacijo z Mealy–evim tipom avtomata. Zapišemo začetno stanje A, v katerem ostajamo toliko časa, dokler se ne začne ena od sekvenc, ki ju zaznavamo. Obe sekvenci se začneta z '1', zato v stanje B preidemo, ko je na vhodu prva '1'. V stanju B ne moremo ostati, saj se na vhodu lahko pojavi '0' ali '1' – v obeh primerih gre za del zaznavanega zaporedja "10X" ali "11X". Iz stanja B preidemo v stanje C, če se vmes pojavi '1', tako da v tem stanju pomeni detekcijo sekvence "11X", v stanje D pa preidemo če se pojavi na vhodu '0', kar pomeni detekcijo sekvence "10X".

Prekrivanje zaporedij: Če se v stanju C pojavi '1' na vhodu, potem gre za sekvenco "111" na vhodu – kar še vedno pomeni, da ostajamo v stanju C, saj je prekrivanje vzorcev dovoljeno. Drugače se diagram obnaša, ko smo v stanju D in pride na vhod še ena '0' – takrat smo imeli na vhodu sekvenco "100", tako da se moramo vrniti v stanje A, saj se nobena od zaznavanih sekvenc ne začneja z '0'.



Delovanje avtomata preizkusimo na testnem zaporedju:

stanje	A	B	D	B	D	A	B	C	D	B	D	A	B	C	D	B	D	B	C	D	A	B	C	D
w	0	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	1	1	0	0	1	1	0
z	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	1

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
16. 12. 2015

1. Določite minimalno konjunktivno normalno obliko (MKNO) in minimalno disjunktivno normalno obliko (MDNO) funkcije  $f$  z uporabo pravil Boole-ove logike in diagramov za minimizacijo funkcij.

$$f(a, b, c, d) = \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + (b \oplus c)) \oplus ((a \uparrow c \uparrow d) \downarrow (\bar{c} + d))}$$

2. Realizirajte podano funkcijo  $f$  z redundancami z eno 4-bitno aritmetično-logično enoto (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

$$f(x_1, x_2, x_3, x_4) = V(1, 4, 5, 8, 9, 12, 15) \quad \text{in} \quad V_x(0, 7, 13, 14)$$

3. Realizirajte funkcijo  $f$  z enim izbiralnikom 4/1.

$$f(a, b, c, d) = \overline{(a \cdot b)} \oplus c \cdot (a + d)$$

4. Pretvorite število  $197_{10}$  ( $11000101_2$ ) v BCD zapis z uporabo "double dabble" algoritma.

## Rešitev 1. naloge

Funkcije NOR ( $\downarrow$ ), NAND ( $\uparrow$ ) in XOR posameznih členov izpišemo. Srednjo ekvivalenco pustimo:

$$\begin{aligned} f(a,b,c,d) &= \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + (b \oplus c)) \oplus ((a \uparrow c \uparrow d) \downarrow (\bar{c} + d))} \\ &= \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}) \oplus ((a \uparrow c \uparrow d) \downarrow (\bar{c} + d))} \\ &= \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}) \oplus ((\overline{a \cdot c \cdot d}) + (\bar{c} + d))} \end{aligned}$$

Nad desnim členom uporabimo De Morgan-ov teorem in ga zapišemo na novo:

$$\begin{aligned} f(a,b,c,d) &= \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}) \oplus (a \cdot c \cdot d \cdot (\bar{c} + d))} \\ &= \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}) \oplus (a \cdot c \cdot d \cdot \bar{c} \cdot \bar{d})} \end{aligned}$$

Uporabimo enakost: ( $\bar{x} \cdot x = 0$ ), zato desni člen postane '0'.

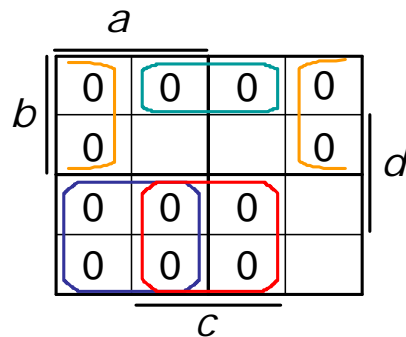
$$f(a,b,c,d) = \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c})} \oplus 0$$

Uporabimo lastnost XNOR operacije ( $\overline{x \oplus 0} = \bar{x}$ ) in dobimo:

$$f = \overline{(a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c})}$$

Negacijo prenesemo na stran  $f$ . Dobimo izraz v DNO, katerega člene vpisujemo kot ničle v Veitch-ev diagram:

$$\bar{f} = a \cdot \bar{b} + b \cdot c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}$$



Iz diagrama lahko zapišemo obliko MDNO tako, da združujemo '1':

$$f_{MDNO} = \bar{a} \cdot \bar{b} \cdot \bar{c} + b \cdot c \cdot d$$

MKNO dobimo tako, da v Veitch-evem diagramu združujemo '0' in dobimo negacijo DNO členov, nakar uporabimo dvakrat De Morgan-ov teorem, da dobljeno negacijo pretvorimo v MKNO:

$$\begin{aligned} \bar{f} &= a \cdot \bar{b} + c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c} \\ f &= \overline{a \cdot \bar{b} + c \cdot \bar{d} + \bar{b} \cdot c + b \cdot \bar{c}} \\ f &= \overline{a \cdot \bar{b} \cdot c \cdot \bar{d} \cdot \bar{b} \cdot c \cdot b \cdot \bar{c}} \\ f_{MKNO} &= (\bar{a} + b) \cdot (\bar{c} + d) \cdot (b + \bar{c}) \cdot (\bar{b} + c) \end{aligned}$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na domači strani predmeta.

## Rešitev 2. naloge:

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji enega tipa. Funkcijo izrišemo v Veitch–ev diagram:

	$x_1$		
$x_2$	1	X	1
	X	1	X
	1		1
	1		X
	$x_3$		$x_4$

Funkcijo zapišemo v obliko DNO, tako da za vse redundance postavimo na '1' razen minterma  $m_{14}$ , ki ga postavimo na '0'.

$$f = \overline{x_3} + x_2 \cdot x_4$$

Dobljeno funkcijo lahko zapišemo v NAND obliki (Sheffer-jev operator), tako da funkcijo dvakrat negiramo:

$$f = \overline{x_3} + x_2 \cdot x_4 = \overline{\overline{\overline{x_3} + x_2 \cdot x_4}} = \overline{\overline{x_3} \cdot \overline{x_2 \cdot x_4}}$$

$$f = x_3 \uparrow (x_2 \uparrow x_4)$$

Druga možna rešitev je v smeri na Piercevega operatorja (NOR):

$$\overline{f} = \overline{x_2} \cdot x_3 + x_3 \cdot \overline{x_4} =$$

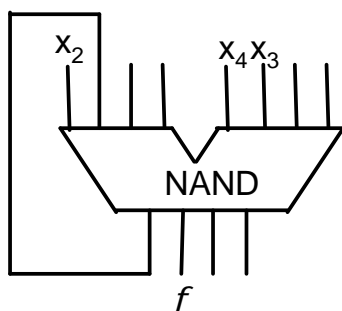
$$f = \overline{\overline{x_2} \cdot x_3 + x_3 \cdot \overline{x_4}}$$

$$f = \overline{\overline{x_2} + \overline{x_3} + x_3 + \overline{x_4}} = (x_2 \downarrow \overline{x_3}) \downarrow (\overline{x_3} \downarrow x_4)$$

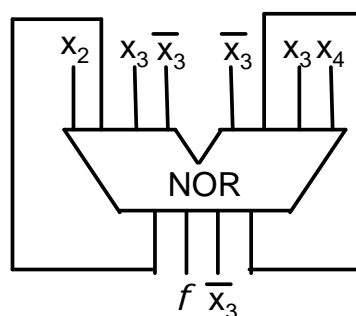
Aritmetično–logično enoto nastavimo tako, da bo izvajala štirikratno dvovhodno NAND operacijo in vhod prvega člena ( $x_2 \uparrow x_4$ ) vežemo nazaj ter tvorimo NAND z  $x_3$ .

Naloga zahteva izvajanje negacij z uporabo ALU, zato negacijo spremenljivke  $x_3$  v primeru realizacije z NOR operatorji izvedemo z uporabo lastnosti:  $\overline{x_3} = x_3 \downarrow 0 = x_3 \downarrow x_3$ .

$$f = x_3 \uparrow (x_2 \uparrow x_4)$$



$$f = (x_2 \downarrow \overline{x_3}) \downarrow (\overline{x_3} \downarrow x_4)$$



### Rešitev 3. naloge:

Funkcija  $f$  je podana v večnivojski (nenormalni) obliki:

$$f(a,b,c,d) = \overline{(a \cdot b)} \oplus c \cdot (a + d)$$

zato jo najprej poenostavimo z uporabo pravil Boole-ove logike. Izpišemo levi člen XOR operacije.

$$f(a,b,c,d) = \overline{(a \cdot b)} \oplus c \cdot (a + d) = (a \cdot b \cdot c + \overline{a \cdot b} \cdot \bar{c}) \cdot (a + d)$$

Uporabimo De Morganov teorem, da dobimo DNO zapis.

$$f(a,b,c,d) = (a \cdot b \cdot c + \overline{a \cdot b} \cdot \bar{c}) \cdot (a + d) = (a \cdot b \cdot c + (\bar{a} + \bar{b}) \cdot \bar{c}) \cdot (a + d)$$

Dobljeni izraz razširimo in vnesemo člen  $a + d$ :

$$\begin{aligned} f(a,b,c,d) &= (a \cdot b \cdot c + (\bar{a} + \bar{b}) \cdot \bar{c}) \cdot (a + d) = \\ &= (a \cdot b \cdot c + \bar{a} \cdot \bar{c} + \bar{b} \cdot \bar{c}) \cdot (a + d) \\ &= a \cdot b \cdot c \cdot (a + d) + \bar{a} \cdot \bar{c} \cdot (a + d) + \bar{b} \cdot \bar{c} \cdot (a + d) \end{aligned}$$

Uporabimo lastnosti Boole-ove logike  $x \cdot \bar{x} = 0$  in  $x \cdot x = x$  in dobimo končni DNO izraz:

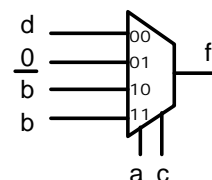
$$\begin{aligned} f(a,b,c,d) &= a \cdot b \cdot c \cdot (a + d) + \bar{a} \cdot \bar{c} \cdot (a + d) + \bar{b} \cdot \bar{c} \cdot (a + d) \\ f(a,b,c,d) &= a \cdot b \cdot c \cdot a + a \cdot b \cdot c \cdot d + \bar{a} \cdot \bar{c} \cdot a + \bar{a} \cdot \bar{c} \cdot d + a \cdot \bar{b} \cdot \bar{c} \cdot a + a \cdot \bar{b} \cdot \bar{c} \cdot d = \\ f(a,b,c,d) &= a \cdot b \cdot c + \bar{a} \cdot \bar{c} \cdot d + a \cdot \bar{b} \cdot \bar{c} \cdot d \end{aligned}$$

Člene dobljene DNO izrišemo v Veitchev diagram:

		$a$			
$b$	$d$	0	1	0	0
		0	1	0	1
	$e$	1	0	0	1
		1	0	0	0
		$c$			

Dobljeni diagram analiziramo po vseh možnih kombinacijah naslovnih vhodov MUX 4/1. Pri vsaki kombinaciji poiščemo samo prvi neprimeren funkcijski ostanek ( $F_{XY}$ ), ki se ga ne da realizirati z eno spremenljivko, tako da opazovano kombinacijo nemudoma ovržemo:

Naslovni vhodi	$F_{00}$	$F_{01}$	$F_{10}$	$F_{11}$
ab	osamljena '1'	osamljena '1'		
ac	<b>d</b>	<b>0</b>	<b>not(b)</b>	<b>b</b>
ad				b xnor c
bc			osamljena '1'	
bd	osamljena '1'			a xnor c
cd				osamljena '1'



Ustrezna kombinacija naslovnih vhodov izbiralnika 4/1 je  $ac$ , funkcijski ostanki so navedeni v zgornji tabeli. Realizacija z enim izbiralnikom 4/1 se nahaja na desni strani tabele.

Rešitev 4. naloge:

Število  $197_{10} = 11000101_2$ . Zapis posameznih števk v BCD zapisu se glasi: 0001 1001 0111.

STOTICE				DESETICE				ENICE												
												1	1	0	0	0	1	0	1	START
										1		1	0	0	0	1	0	1		POMIK1
									1	1		0	0	0	1	0	1			POMIK2
									1	1	0	0	0	1	0	1				POMIK3
								1	0	0	1	0	0	1	0	1				ADD3
							1	0	0	1	0	0	1	0	1					POMIK4
						1	0	0	1	0	0	1	0	1						POMIK5
					1	0	0	1	0	0	1	0	1							ADD3
					1	0	0	1	1	0	0	0	1							POMIK6
				1	0	0	1	1	0	0	0	1								ADD3
				1	1	0	0	1	0	1	1	1								POMIK7
			1	1	0	0	1	0	1	1	1									POMIK8
$1_{10}$				$9_{10}$				$7_{10}$												

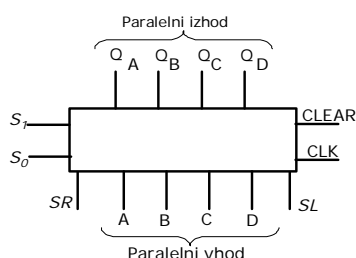
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na domači strani predmeta.



# RAZVOJ DIGITALNIH SISTEMOV

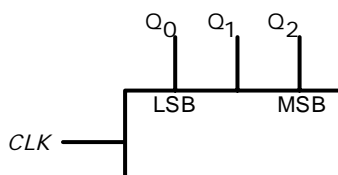
2. kolokvij  
22. 1. 2016

1. Realizirajte JK-flip flop z uporabo D- flip flopa in logičnih vrat.
2. Z uporabo D flip-flopov in izbiralnikov 4/1 prikažite sintezo univerzalnega 4-bitnega pomikalnega registra, ki ima dva funkcijska vhoda  $S_0$  in  $S_1$  in opravlja funkcije po spodnji tabeli. Register ima tudi zaporedna vhoda za pomik v levo (SL – serial left), pomik v desno (SR – serial right) in asinhroni vhod za brisanje CLEAR (aktiven nizek).

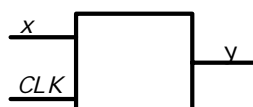


$S_1$	$S_0$	<i>funkcija</i>
0	0	drži stanje
0	1	pomik vsebine eno mesto desno
1	0	pomik vsebine eno mesto levo
1	1	nalaga vsebino z vhodov ABCD

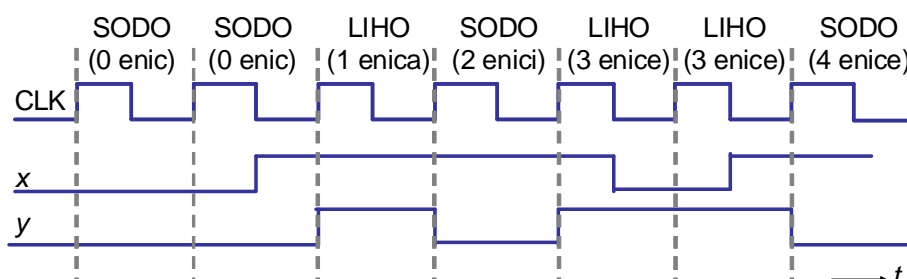
3. Prikažite sintezo sinhronega 3-bitnega števca navzdol z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov ter vezje narišite. Imena signalov so razvidna iz spodnje slike.



4. Realizirajte generator lihe parnosti (paritete) kot Moore-ov avtomat končnih stanj, ki šteje število enic v serijskem zaporedju bitov  $x$  na vhodu: Izhod vezja  $y$  naj bo '1', ko je na vhodu liho število enic in '0' ko je na vhodu sodo število enic. Ob resetu avtomata je število enic na vhodu sodo (nič enic). Za realizacijo uporabite D flip-flope, prožene na sprednji rob signala ure  $CLK$ .



Primer delovanja generatorja parnosti povzema spodnja slika:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 1. naloge

Za vezje JK-FF narišemo pravilnostno tabelo, pri čemer na vhodni strani zberemo vhode J, K in trenutno stanje  $Q(t)$ , na izhodni pa naslednje stanje  $Q(t+1)$ . JK-FF opravlja štiri funkcije (HOLD, RESET, SET, INVERT) glede na kombinacijo vhodnih signalov, medtem ko D-FF opravlja samo dve (SET, RESET).

$J$	$K$	$Q(t)$	$Q(t+1)$	$D$	<i>funkcija D</i>	<i>funkcija JK</i>
0	0	0	0	0	RESET	HOLD
0	0	1	1	1	SET	HOLD
0	1	0	0	0	RESET	RESET
0	1	1	0	0	RESET	RESET
1	0	0	1	1	SET	SET
1	0	1	1	1	SET	SET
1	1	0	1	1	SET	INVERT
1	1	1	0	0	RESET	INVERT

Iz tabele narišemo Veitchev diagram za vhod D v odvisnosti od vhodov J, K in trenutnega stanja  $Q(t)$ .

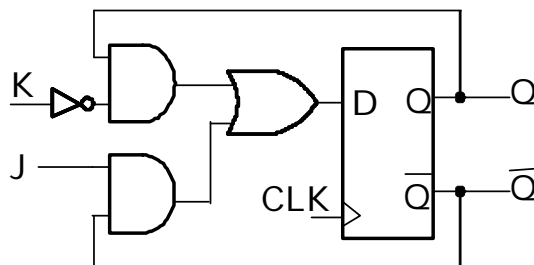
$D$ :

	$J$		
$K$	1	0	0
	1	1	1
	$Q(t)$		

Dobljeni diagram minimiziramo in zapišemo enačbo vhoda D:

$$D = J \cdot \overline{Q(t)} + \bar{K} \cdot Q(t)$$

Vezje narišemo:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

## Rešitev 2. naloge:

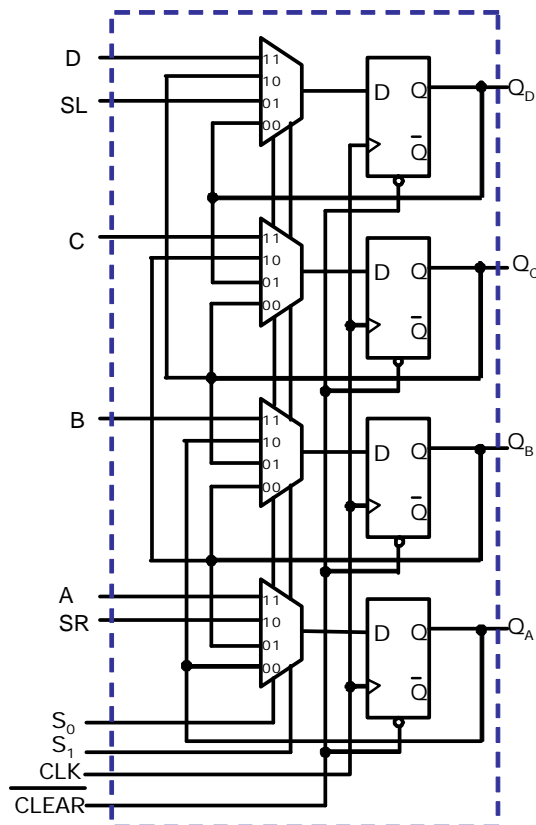
Vsako od operacij izpišemo v pravilnostno tabelo v kateri združimo funkcijska bita  $S_1$  in  $S_0$  in trenutno stanje na  $i$ -tem mestu registra  $Q_i(t)$ . Mesta registra od leve proti desni so  $Q_i = (Q_A, Q_B, Q_C, Q_D)$ . Realizacija z D flip-flopi nam analizo močno poenostavi, zaradi enačbe D flip-flopa:  $D = Q(t+1)$ .

Tabela 1: Prehajanje stanj univerzalnega registra.

$S_1$	$S_0$	$Q_i(t+1)$	funkcija
0	0	$Q_i(t)$	HOLD
0	1	$Q_{i+1}(t)$	LSR
1	0	$Q_{i-1}(t)$	LSL
1	1	$x_i$	LOAD

Iz poenostavljene tabele prehajanja stanj univerzalnega registra sestavimo realizacijo, ki bo vključevala izbiralnike MUX 4/1 in D-FF.

Na naslovna vhoda vseh MUX 4/1 vodimo funkcijska signala  $S_1$  in  $S_0$ . Potem na vsakem podatkovnem vhodu realiziramo ustrezno funkcijo.



Stanje  $S_1S_0 = "00"$  pomeni držanje stanja (*HOLD*), torej bodo trenutne vrednosti D-FF ohranile vrednost  $Q_i(t+1) = Q_i(t)$ . Na sliki to realiziramo tako, da vodimo izhod D-FF nazaj na vhod pri podatkovnem vhodu 00. Stanje  $S_1S_0 = "01"$  pomeni pomik desno (*LSR* – ang. logic shift right), torej bodo D-FF pomaknili vsebino eno mesto desno. Pomik desno pomeni, da na mesto skrajno levega bita vpišemo vrednost zaporednega vhoda SR, nato  $Q_A$  vodimo na vhod  $Q_B$  in tako do skrajno desnega bita. Stanje  $S_1S_0 = "10"$  pomeni pomik levo (*SHL* – ang. shift left), torej bodo D-FF pomaknili vsebino eno mesto levo. Pomik levo pomeni, da na mesto skrajno desnega bita vpišemo vrednost zaporednega vhoda SL, nato  $Q_D$  vodimo na vhod  $Q_C$  in tako do skrajno levega bita.  $S_1S_0 = "11"$  pomeni vzporedno nalaganje z vhodov (*LOAD*)  $Q_D(t+1) = D$ ,  $Q_C(t+1) = C$ ,  $Q_B(t+1) = B$ ,  $Q_A(t+1) = A$ . Na tabeli smo  $i$ -ti vhod za vzporedno nalaganje označili kot  $x_i = (A, B, C, D)$ .

### Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števec:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch-eve diagrame za tri spremenljivke za vsak vhod T-FF, vendar ker so T-FF po svoji naravi primerni za realizacijo števec, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Iz stolpca T<sub>0</sub> se vidi:

$$T_0 = 1$$

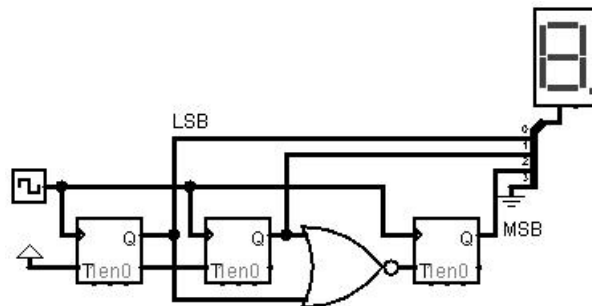
Z opazovanjem stolpcev trenutnega stanja določimo T<sub>1</sub>:

$$T_1 = \overline{Q_0}$$

Podobno lahko določimo T<sub>2</sub>:

$$T_2 = \overline{Q_0} \cdot \overline{Q_1} = \overline{Q_0 + Q_1}$$

Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\ counter\_7\_0\_using\_T\_FF.circ:



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

#### Rešitev 4. naloge:

Postopek sinteze zahteva, da realiziramo avtomat končnih stanj Moore-ove izvedbe: Najprej bomo razvili diagram prehajanja stanj, ki opisuje delovanje vezja za liho preverjanje parnosti. Vezje je lahko v enem od dveh stanj: v sekvenci je bilo do tega trenutka liho ali sodo število enic. Kadar je na vhodu 1, je potrebno preklopiti v drugo stanje. Na primer, če je bilo do tega trenutka prisotnih liho število enic in je trenutni vhod 1, potem bomo imeli sedaj sodo število enic. Če pa bo na vhodu 0, ostane v istem stanju. Narisani diagram prehajanja stanj ima dve stanji, ki označujeta trenutno število enic na vhodu – torej LIHO in SODO. Izhod zapišemo pod stanjem (LIHO='1', SODO='0'). Vrednosti na vhodu  $x$  povzročajo spreminjanje stanj, ki so označene z usmerjenimi povezavami. Če je se na vhodu pojavi '0' (ne glede na to v katerem stanju smo) ostanemo v tem stanju: Jasno – saj štejemo samo '1'. Če smo v stanju LIHO in se na vhodu pojavi '1', preidemo v SODO. Če smo v stanju SODO in se na vhodu pojavi '1', preidemo v LIHO. Povedano povzema spodnji diagram prehajanja stanj

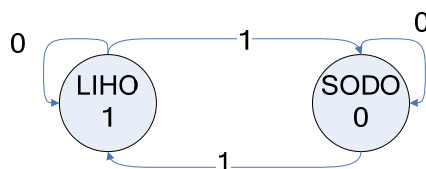


Diagram prehajanja stanj opišemo s tabelo prehajanja stanj:

vhod $x$	trenutno stanje $Q(t)$	naslednje stanje $Q(t+1)$
0	SODO	SODO
0	LIHO	LIHO
1	SODO	LIHO
1	LIHO	SODO

$x$	$Q(t)$	$Q(t+1)$	$D$	$y$
0	0	0	0	0
0	1	1	1	0
1	0	1	1	1
1	1	0	0	1

Iz tabele prehajanja stanj avtomata določimo enačbo za D-FF. Potrebno število FF je 1, saj sta stanji samo dve.

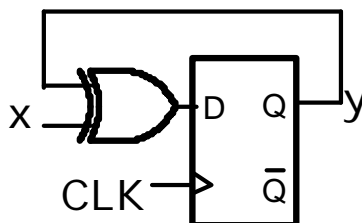
Iz aplikacijske tabele sledi:

$$D = Q(t+1)$$

$$Q(t+1) = x \cdot \overline{Q(t)} + \overline{x} \cdot Q(t) = x \oplus Q(t)$$

$$y = Q(t)$$

Izvedba vezja je:



Realizirali smo T-FF, prožen na sprednji rob signala ure.

Delovanje vezja si lahko ogledate v predlogah Logisim na domači strani predmeta:

Logisim\ff\T\_ff\_using\_D\_ff\_and\_xor.circ

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
12. 12. 2016

1. Izrazite podano logično funkcijo v konjunktivni normalni obliki samo s Shefferjevimi operatorji. Morebitne negacije realizirajte s Shefferjevim operatorjem.

$$f(a,b,c) = (a+b) \cdot (\bar{b} + \bar{c}) \cdot c$$

2. Določite minimalno normalno obliko (MNO) funkcije  $f$  z uporabo diagramov za minimizacijo funkcij in COST funkcije. Dobljeno obliko MNO realizirajte z eno 4-bitno aritmetično-logično enoto (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

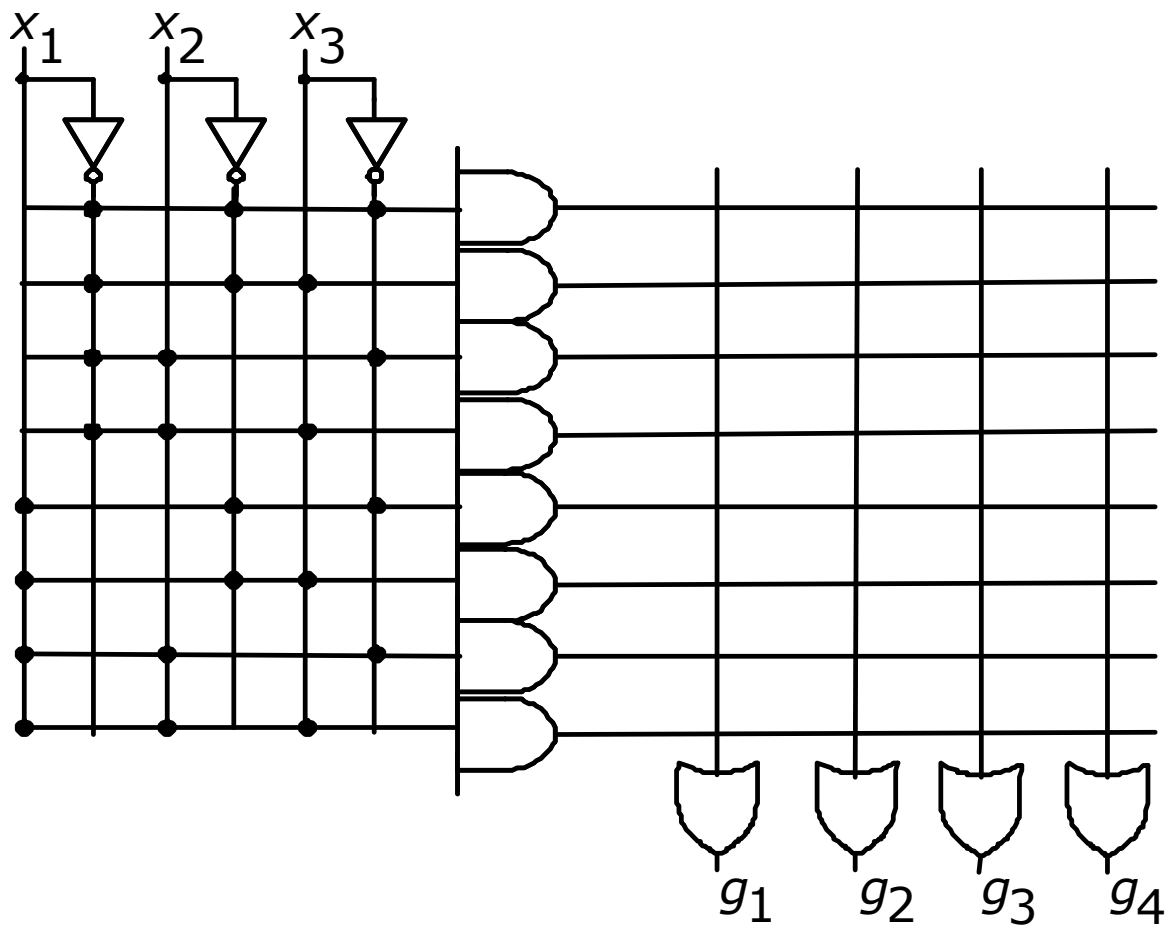
$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5, 6, 9, 10, 14) \text{ in } \&_x(0, 4, 7, 11, 13)$$

3. Programirajte ROM vezje za realizacijo naslednjih funkcij:

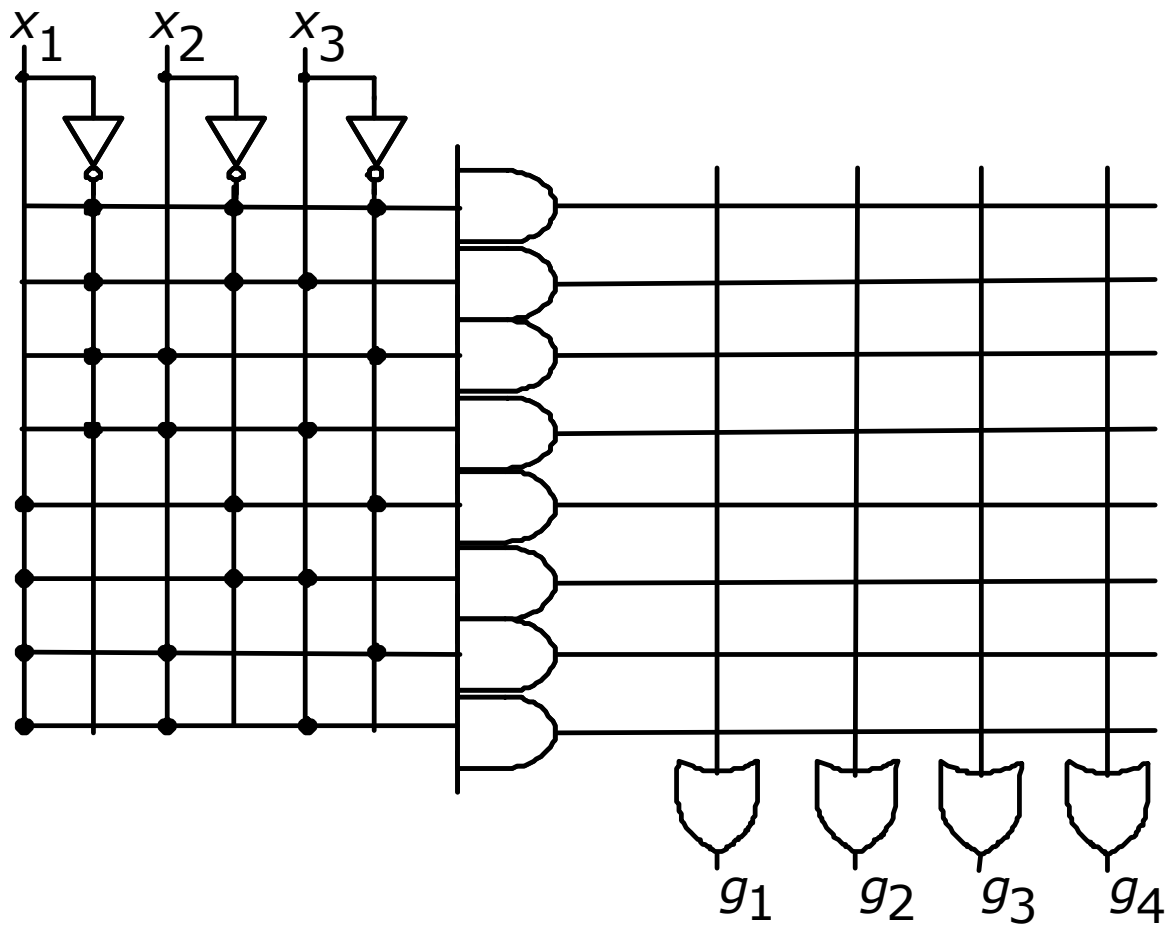
$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

ROM vezje ima 3 vhodne spremenljivke in 4 bitno vsebino. Povezave oz. 'varovalke' označite s piko (●). Uporabite shemo na hrbtni strani.

4. Pretvorite število  $2017_{10}$  ( $7E1_{16}$ ) v BCD zapis z uporabo "double dabble" algoritma.



Če se zmotite, uporabite spodnjo shemo - ne obeh!



# DEVELOPMENT OF DIGITAL SYSTEMS

Midterm Examination

12. 12. 2016

1. Implement a given function  $f$ , expressed in a conjunctive form below, using only Sheffer operators. All possible variable negations have to be implemented using Sheffer operator.

$$f(a, b, c) = (a + b) \cdot (\bar{b} + \bar{c}) \cdot c$$

2. Write the minimal normal form (MNO) of a given function  $f$ , using Veitch minimization diagrams and COST function. Implement this minimal normal form using a single 4-bit arithmetic logic unit (ALU). All possible variable negations have to be implemented using this ALU.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5, 6, 9, 10, 14) \text{ in } \&_x(0, 4, 7, 11, 13)$$

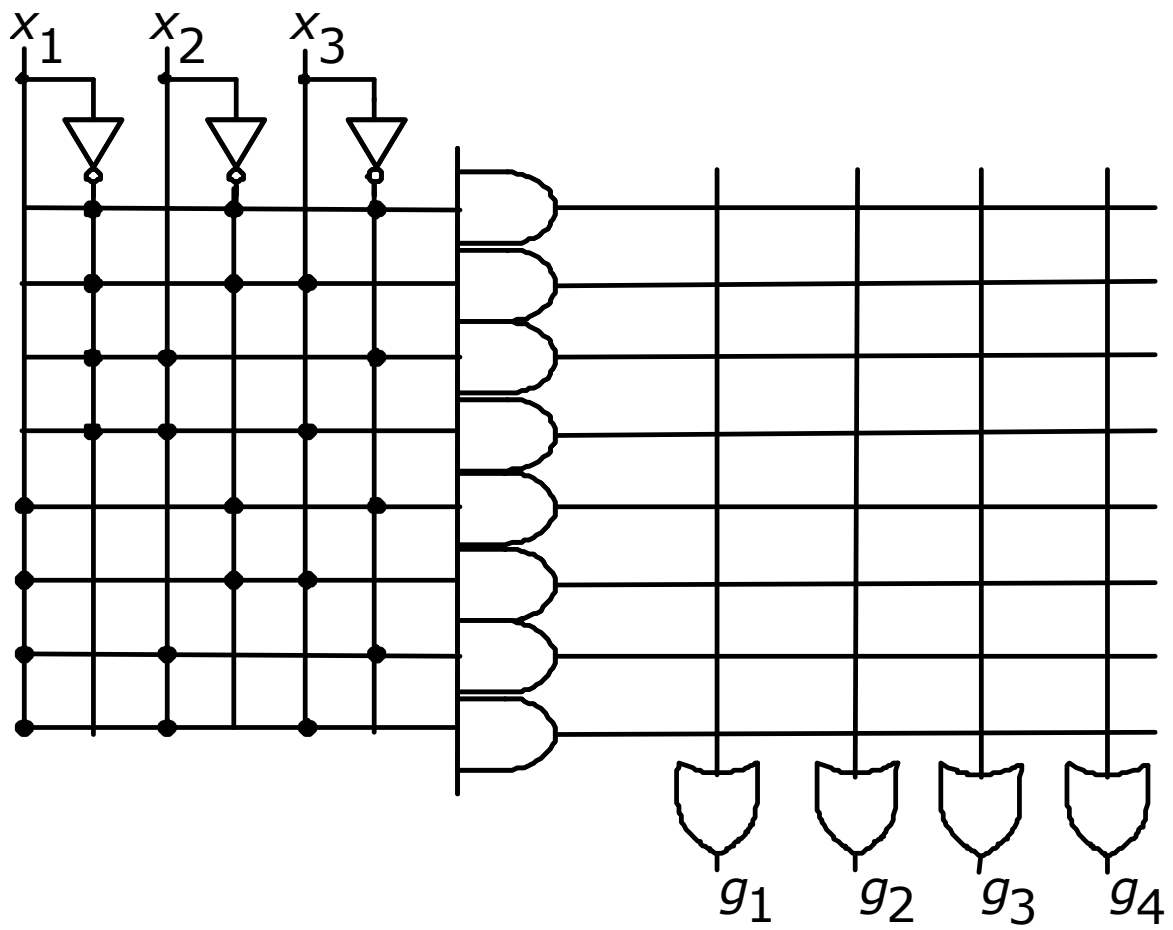
3. Program a ROM element, which implements following functions:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

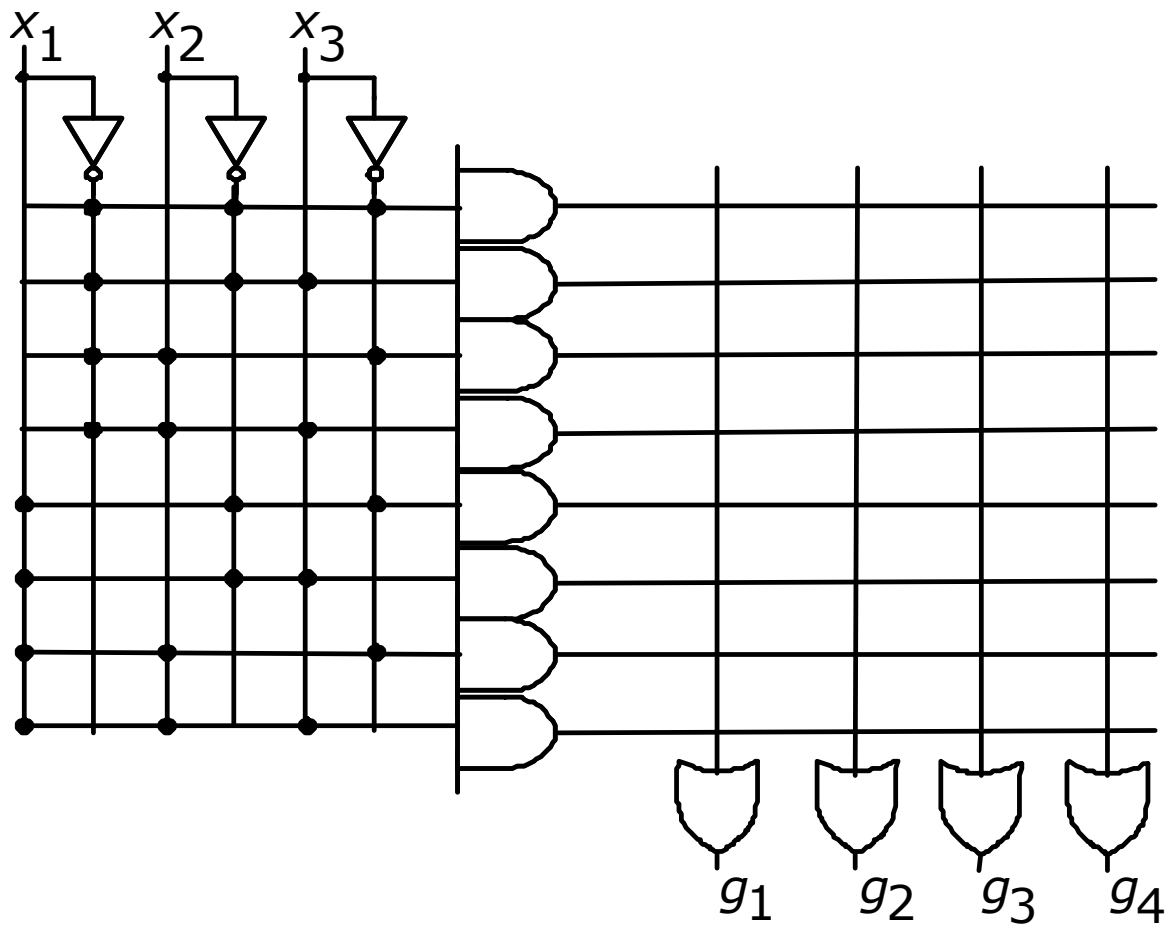
The ROM element has 3 inputs and 4-bit contents. Mark the programmed connections using a dot symbol (●). Use the provided schematic on the back side.

4. Convert the decimal number  $2017_{10}$  ( $7E1_{16}$ ) into its BCD representation using "double dabble" algorithm.





Upon error, use the scheme below - not both!



## Rešitev 1. naloge

Shefferjeva normalna oblika zahteva realizacijo s samimi [NAND](#) ( $\uparrow$ ) operatorji:

$$f(a, b, c) = (a + b) \cdot (\bar{b} + \bar{c}) \cdot c$$

Celotno funkcijo negiramo dvakrat in obenem negiramo dvakrat tudi vse disjunktivne člene (OR) ter zapišemo na novo:

$$f(a, b, c) = \overline{\overline{(a + b)} \cdot \overline{(\bar{b} + \bar{c})} \cdot c}$$

Nad disjunktivnima členoma uporabimo De Morganov teorem, tako da iz teh členov nastanejo konjukcije (AND). Preostala negacija nad členom skupaj s konjunkcijo tvori NAND operator.

$$f(a, b, c) = \overline{(\bar{a} \cdot \bar{b}) \cdot (\bar{b} \cdot \bar{c}) \cdot c}$$

Iz nastale funkcije samo še preberemo NAND operatorje:

$$f(a, b, c) = \overline{(\bar{a} \uparrow \bar{b}) \uparrow (b \uparrow c) \uparrow c}$$

Negacije v nastalem izrazu izpišemo, pri čemer uporabimo lastnost Shefferjevih operatorjev ( $\bar{x} = x \uparrow x$  oziroma  $\bar{x} = 1 \uparrow x$ ) in dobimo končni rezultat:

$$f(a, b, c) = (((a \uparrow a) \uparrow (b \uparrow b)) \uparrow (b \uparrow c) \uparrow c) \cdot (((a \uparrow a) \uparrow (b \uparrow b)) \uparrow (b \uparrow c) \uparrow c)$$

Enostavnejšo in cenejšo (COST) rešitev dobimo, če funkcijo predhodno poenostavimo z uporabo pravil Boole-ove algebre, pri čemer velja ( $x \cdot \bar{x} = 0$  in  $x \cdot x = x$ ):

$$f(a, b, c) = (a + b) \cdot (\bar{b} + \bar{c}) \cdot c = (a \cdot c + b \cdot c) \cdot \bar{b} \cdot c = a \cdot \bar{b} \cdot c$$

Dobljena funkcija ima samo en člen, nad katerim izvedemo dvojno negacijo, eno uporabimo za izražavo s Sheffer-jevimi operatorji, drugo pa izrazimo kot ( $\bar{x} = x \uparrow x$  oziroma  $\bar{x} = 1 \uparrow x$ ):

$$\begin{aligned} f(a, b, c) &= \overline{a \cdot \bar{b} \cdot c} = \overline{a \uparrow \bar{b} \uparrow c} = \overline{a \uparrow (b \uparrow b) \uparrow c} \\ f(a, b, c) &= (a \uparrow (b \uparrow b) \uparrow c) \uparrow (a \uparrow (b \uparrow b) \uparrow c) \end{aligned}$$

## Rešitev 2. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami:

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5, 6, 9, 10, 14) \text{ in } \&_x(0, 4, 7, 11, 13)$$

Najprej jo pretvorimo v obliko PDNO, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	X
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 8, 11, 15)$$

Funkcijo minimiziramo, zapišemo v MDNO in MKNO ter poiščemo MNO.

	$x_1$				
$x_2$	1	0	0	X	$x_4$
	0	X	1	0	
	0	X	1	0	
	X	0	X	1	
	$x_3$				

$$f_{MDNO} = x_3 \cdot x_4 + \overline{x_3} \cdot \overline{x_4}$$

Podobno storimo še za MKNO.

		$x_1$			
$x_2$		1	0	0	X
		0	X	1	0
		0	X	1	0
		X	0	X	1
		$x_3$			

$$\overline{f_{MKNO}} = \overline{x_3 \cdot x_4 + \overline{x_3} \cdot \overline{x_4}}$$

$$f_{MKNO} = \overline{x_3 \cdot x_4 + \overline{x_3} \cdot \overline{x_4}}$$

$$f_{MKNO} = (\overline{x_3 \cdot x_4}) \cdot (\overline{\overline{x_3} \cdot \overline{x_4}})$$

$$f_{MKNO} = (\overline{x_3} + x_4) \cdot (x_3 + \overline{x_4})$$

V obeh realizacijah preštejemo operatorje ter število vhodov, ter rezultate povzamemo v spodnji tabeli

Tabela 1: COST funkcija.

COST	VRAT	VHODOV
MDNO	3	6
MKNO	3	6

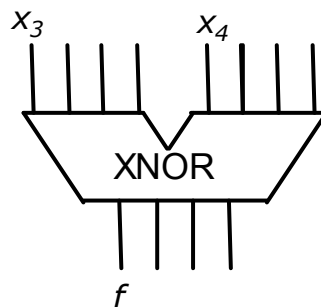
Iz tabele sledi: MNO=MDNO=MKNO.

Aritmetično–logično enota lahko poleg aritmetičnih naenkrat realizira štiri dvovhodne logične operacije *istega tipa* (OR, AND, NOT, NOR, NAND, XOR, XNOR), zato nas zanima realizacija zgornje funkcije z dvovhodnimi operatorji *enega tipa*. Pri realizaciji so zato primerne čimbolj nenormalne oblike (večnivojske oblike), samo da vsebujejo operatorje ene vrste (Sheffer, Pierce, linearne funkcije).

Podana funkcija v MNO zato za neposredno realizacijo s 4-bitno ALU ni primerna. Enostavno jo prevedemo na operator enega tipa, če izrazimo MDNO ali MKNO, saj gre v obeh primerih za XNOR operacijo:

$$f_{MDNO} = f_{MKNO} = x_3 \cdot x_4 + \overline{x_3} \cdot \overline{x_4} = \overline{x_3 \cdot x_4 + \overline{x_3} \cdot \overline{x_4}} = \overline{x_3 \oplus x_4} = x_3 \oplus x_4 = (x_3 \equiv x_4)$$

Aritmetično logično enoto nastavimo tako, da opravlja 4-bitno XNOR operacijo. Na vhoda npr. prvega mesta postavimo  $x_3$  in  $x_4$  ter na prvem izhodu dobimo  $f$ .



Rešitev 3. naloge:

Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ev diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$\begin{aligned}g_1(x_1, x_2, x_3) &= x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \\g_1(x_1, x_2, x_3) &= V(4, 6, 5, 7, 0)\end{aligned}$$

Podobno storimo še za preostale funkcije:

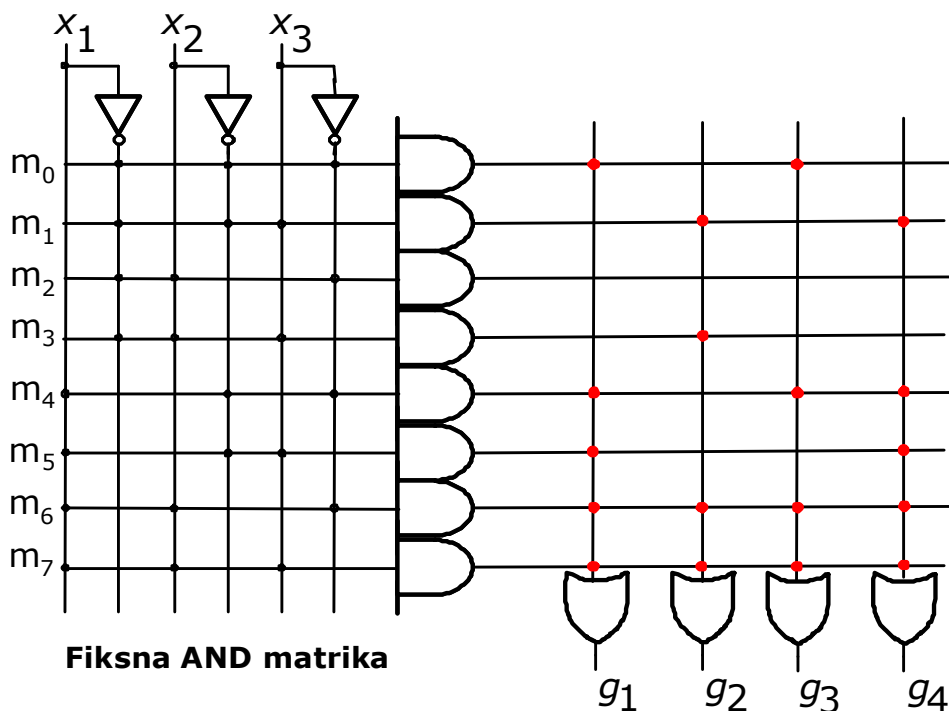
$$\begin{aligned}g_2 &= \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) \\g_2(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\g_2(x_1, x_2, x_3) &= V(1, 3, 6, 7)\end{aligned}$$

$$\begin{aligned}g_3 &= \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 = \\g_3(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) = \\g_3(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\g_3(x_1, x_2, x_3) &= V(0, 4, 6, 7)\end{aligned}$$

$$\begin{aligned}g_4(x_1, x_2, x_3) &= \overline{x_2} \cdot x_3 + x_1 \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 \\g_4(x_1, x_2, x_3) &= V(1, 5, 4, 6, 7)\end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ( $x_1, x_2, x_3$ ) od  $m_0$  do  $m_7$ . Številka minterma določa naslov lokacije ROM pomnilnika.

Narišemo celotno vezje ROM strukture in vstavimo pike (•) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk. Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete. Rezultati bodo objavljeni na domači strani predmeta.

Vsebino ROM pomnilnika kratko opišemo s tabelo v kateri programirane povezave (●) v OR matriki pišemo kot '1' na danem mestu vsebine.

<i>Minterm</i> $g_i(x_1x_2x_3)$	<i>Naslov lokacije</i> $x_1x_2x_3$	<i>Vsebina</i> <i>lokacije</i> $g_1g_2g_3g_4$
$m_0$	$000_2$	$1010_2$
$m_1$	$001_2$	$0101_2$
$m_2$	$010_2$	$0000_2$
$m_3$	$011_2$	$0100_2$
$m_4$	$100_2$	$1011_2$
$m_5$	$101_2$	$1001_2$
$m_6$	$110_2$	$1111_2$
$m_7$	$111_2$	$1111_2$

Realni ROM elementi imajo 8 bitne podatke, zloge ali včasih oktete (ang. *byte, octet*). Vsebina ROM elementov se podaja v datoteki, ki jo nato programiramo z posebnim inštrumentom (ROM programatorjem).

Najenostavnejši način podajanja zapisa vsebine ROM elementa je v surovi dvojiški obliki (ang. *raw binary file*) v kateri si 8 bitni podatki sledijo zapisani v dvojiški obliki. Kompleksnejša zapisa podajanja vsebine ROM s tabelo sta INTEL šestnajstiška oblika (ang. [\*Intel hex record\*](#)) in Motorola SREC oblika (ang. [\*Motorola S record\*](#)).

Rešitev 4. naloge:

Število  $2017_{10} = 111\ 1110\ 0001_2$ .

Zapis posameznih števk v BCD zapisu se glasi: 0010 0000 0001 0111. Prva pomika smo v spodnji shemi preskočili, ker do prištevanja ne more priti.

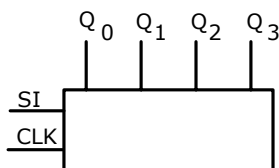
TISOČICE		STOTICE		DESETICE		ENICE		Dvojiško število										
								1	1	1	1	1	1	0	0	0	0	1
							1	1	1	1	1	0	0	0	0	1		
							1	1	1	1	0	0	0	0	1			
							1	0	1	0								
					1		0	1	0	1	1	1	0	0	0	1		
							1	0	0	0								
					1	1	0	0	0	0	1	1	0	0	0	1		
					1	1	0	0	0	1	1	0	0	0	1			
				1	0	0	1											
		1		0	0	1	0	0	1	1	0	0	0	0	1			
								1	0	0	1							
			1	0		0	1	0	0	1	0	0	0	1				
					1	0	0	0										
			1	0	1	0	0	0	0	1	0	0	0	1				
								0	1	0	0							
								1	0	0	0							
								0	0	1	1							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0	1	0	0							
								0	0	1	0							
								0										

# RAZVOJ DIGITALNIH SISTEMOV

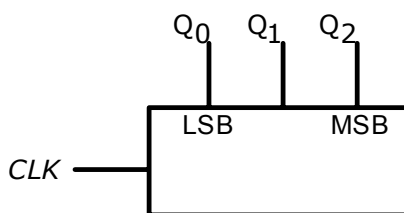
2. kolokvij

27. 1. 2017

1. Realizirajte JK-flip flop z uporabo T- flip flopa in logičnih vrat.
2. Narišite vezje 4-bitnega pomikalnega registra z JK celicami in logičnimi vrati. Register ima zaporedni vhod *SI* (ang. serial input) in vzporedni izhod ( $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ ).



3. Prikažite sintezo sinhronega števca, ki šteje po Grayevi kodi navzdol z uporabo T flip-flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip-flopov ter vezje narišite. Imena signalov so razvidna iz spodnje slike. Števno zaporedje po Grayevi kodi navzdol:  $4 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 0 \rightarrow 4 \rightarrow 5 \dots$



4. Načrtajte diagram stanj Moore-ovega avtomata končnih stanj, ki krmili delovanje garažnih vrat: Garažna vrata imajo vhod VRATA ter vhod ZAŠČITA, ki postane '1' vedno, ko preko motorja steče dovolj velik tok. Z meritvijo toka na motorju obenem izdelamo funkcijo detekcije obeh končnih položajev, kot tudi zaščito proti oviram na poti vrat. Vezje ima 2-bitni izhod za enosmerni motor:

Koda operacije		Funkcija izhoda
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol. Podobno je v obratni smeri: Če na poti naletijo na oviro ali pridejo do spodnje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo pomikati v smeri navzgor.

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni domači strani predmeta.

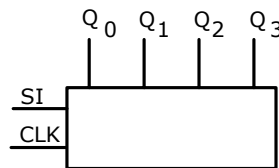


# DEVELOPMENT OF DIGITAL SYSTEMS

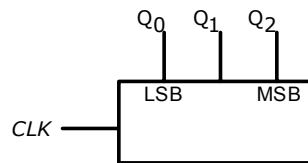
Midterm examination

27. 01. 2017

1. Implement a JK-flip flop using a T- flip flop and logic gates.
2. Draw the circuit diagram of a four bit shift SIPO register, using JK type flip-flops and any logical gates. The SIPO register has a serial input  $SI$  (ang. serial input) and a parallel output ( $Q_0, Q_1, Q_2, Q_3$ ). Name the signals according to figure below.



3. Synthesize a 3 bit Gray down counter using T type flip-flops: Draw the state transition table, determine T flip-flop input equations and draw the resulting counter circuit using signal names in the figure below.  
Note: 3-bit Gray down counting sequence:  $4 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 0 \rightarrow 4 \rightarrow 5 \dots$



4. Draw the state transition diagram of a Moore type finite state machine, which controls the movement of a garage door. The control circuit has an input (DOOR) and an input (PROTECTION), which is set to '1' whenever a current limit of the motor is reached. The motor current limit input is used to detect both door end positions as well as for protection against obstacles in the door path. The control circuit has a two bit motor output:

Operation code		Motor operation
OP <sub>1</sub>	OP <sub>0</sub>	
0	0	motor stop
0	1	door moving upwards
1	0	door moving downwards

When the door knob is pressed ( $DOOR=1$ ), the door starts to move upwards. If an obstacle is in the door path or the door reaches its upper end position ( $PROTECTION=1$ ), the motor stops. When the door knob is pressed again, the door starts to move downwards until the protection limit is reached again. After the door knob is pressed again, the process is repeated.

## Rešitev 1. naloge

Za vezje JK-FF narišemo pravilnostno tabelo, pri čemer na vhodni strani zberemo vhode J, K in trenutno stanje  $Q(t)$ , na izhodni pa naslednje stanje  $Q(t+1)$ . JK-FF opravlja štiri funkcije (HOLD, RESET, SET, INVERT) glede na kombinacijo vhodnih signalov, medtem ko T-FF opravlja samo dve (HOLD, INVERT).

$J$	$K$	$Q(t)$	$Q(t+1)$	$T$	funkcija $T$	funkcija $JK$
0	0	0	0	0	HOLD	HOLD
0	0	1	1	0	HOLD	HOLD
0	1	0	0	0	HOLD	RESET
0	1	1	0	1	INVERT	RESET
1	0	0	1	1	INVERT	SET
1	0	1	1	0	HOLD	SET
1	1	0	1	1	INVERT	INVERT
1	1	1	0	1	INVERT	INVERT

Iz tabele narišemo Veitchev diagram za vhod T v odvisnosti od vhodov J, K in trenutnega stanja  $Q(t)$ .

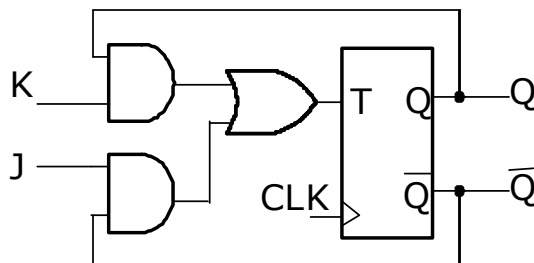
$T:$

	$J$			
$K$	1	1	1	0
	1	0	0	0
	$Q(t)$			

Dobljeni diagram minimiziramo in zapišemo enačbo vhoda T:

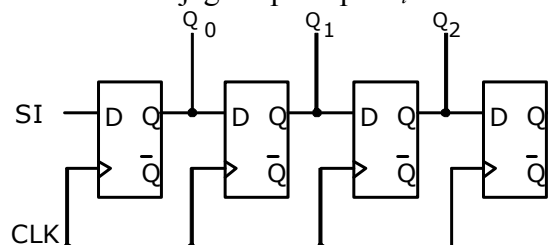
$$T = J \cdot \overline{Q(t)} + K \cdot Q(t)$$

Vezje narišemo:



## Rešitev 2. naloge:

Zaporedno-vzporedni (SIPO) pomikalni register, realiziran s pomočjo D-FF, je veriga kaskadno vezanih D-FF, v kateri je izhod prejšnjega flip-flopa  $Q_{i-1}$  vezan na vhod naslednjega flip-flopa  $D_i$ .



Če želimo pomikalni register sestaviti iz JK-FF in logičnih vrat, je en način realizacije realizirati celico D-FF s pomočjo JK-FF in logičnih vrat.

V ta namen zapišemo tabelo D-FF, pri kateri dodamo izhodni stolpec JK

vhodov.

$D$	$Q(t)$	$Q(t+1)$	$J$	$K$	Pomen stanja
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

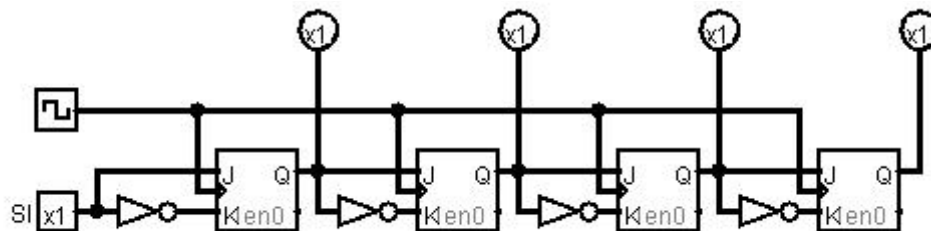
Iz tabele izrazimo J in K vhoda s pomočjo D vhoda in trenutnega stanja  $Q(t)$ . Če redundance (X) postavimo primerno, dobimo:

$$J = D$$

$$K = \bar{D}$$

Če nastali D-FF sestavimo skupaj v 4-bitni pomikalni register dobimo spodnjo realizacijo.

Ena možna rešitev je opisana realizacija D-FF z JK-FF, ki je prikazana na spodnji sliki.

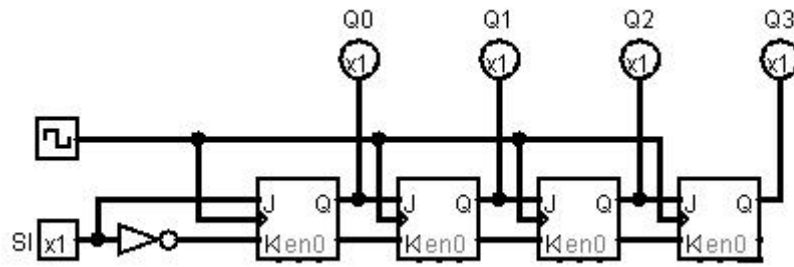


Drugo možnost predstavlja neposredna realizacija s pomočjo tabele pomikanja med opazovanim mestom  $Q_i$  in naslednjim mestom  $Q_{i+1}$  pomikalnega registra. Ponovno zapišemo vzbujalno tabelo za opazovano mesto in določimo vhode glede na spremembo stanja ( $Q_{i+1}(t) \rightarrow Q_{i+1}(t+1)$ ) na  $(i+1)$  mestu.

Trenutna vsebina registra		Vsebinska registra ob pomiku	Vhoda FF na mestu $i+1$		Pomen stanja
$Q_i(t)$	$Q_{i+1}(t)$	$Q_{i+1}(t+1)$	$J_{i+1}$	$K_{i+1}$	
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

Če v dobljenih vseh za mesto  $(i+1)$  izberemo primerne redundance, dobimo vhoda  $J_{i+1} = Q_i(t)$  in  $K_{i+1} = Q_i(t)'$ . To velja tudi za vhodno mesto, torej bomo na vhodni JK-FF vezali vhod  $J_0 = x(t)$  in  $K_0 = x(t)'$ .

Neposredna realizacija pomikalnega registra z JK–FF je prikazana na spodnji sliki.



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:  
Logisim\shift\_reg\shift\_reg\_4bit\_using\_jkff.circ

Rešitev 3. naloge:

Desetiška števna sekvenca po 3-bitni Grayevi kodi navzdol se glasi: ...  
4→5→7→6→2→3→1→0→4→5 ...

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

Trenutno stanje			Naslednje stanje			Enačbe FF		
Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1
0	1	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch-eve diagrame za tri spremenljivke za vsak vhod T-FF, vendar ker so T-FF po svoji naravi primerni za realizacijo števcov, so praviloma njihove vhodne enačbe zelo

enostavne. Iz tabele prehajanja stanj števca določimo enačbe T-FF:

Za vhod T<sub>0</sub> narišemo Veitchev diagram:

$T_0$ :

	$Q_2$			
$Q_1$	0	1	0	1
	1	0	1	0
	$Q_0$			

Funkcija na Veitchevem diagramu je linearna - če izračunamo koeficiente dobimo:

$$T_0 = Q_2 \oplus Q_1 \oplus Q_0$$

Če izpišemo vrednost T<sub>1</sub>, dobimo:

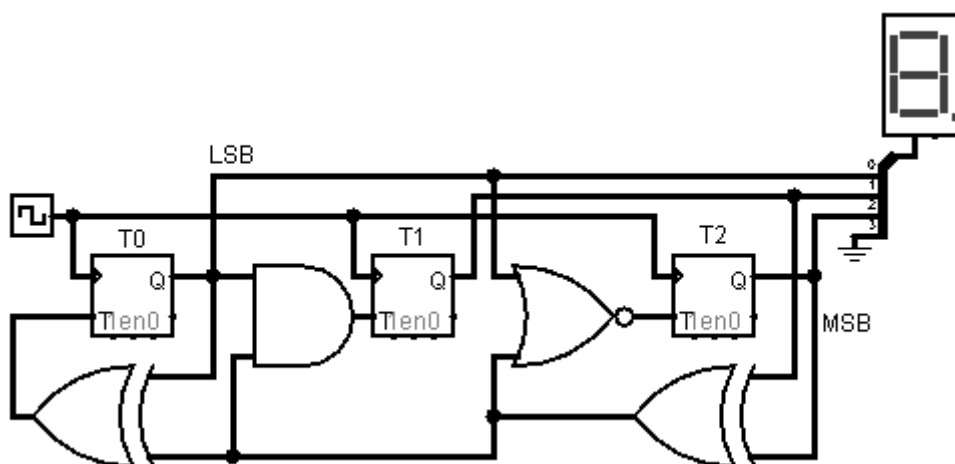
$$T_1 = \overline{Q_2} \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q_1} \cdot Q_0 = (Q_2 \oplus Q_1) \cdot Q_0$$

Podobno lahko določimo T<sub>2</sub>:

$$T_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot Q_1 \cdot \overline{Q_0} = (\overline{Q_2 \oplus Q_1}) \cdot \overline{Q_0}$$

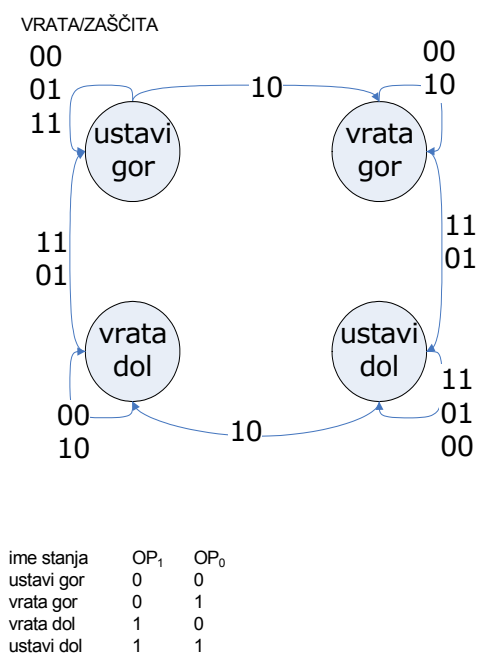
$$T_2 = (\overline{Q_2 \oplus Q_1}) + Q_0$$

Opis delovanja in vezje števca je v predlogah vaj na domači strani predmeta v imeniku Logisim\counter\counter\_down\_3\_bit\_Gray\_using\_T\_FF.circ:



#### Rešitev 4. naloge:

Narišemo Moore-ov diagram stanj:



Iz opisa naloge je razvidno, da stanje "ustavi" ni samo eno, ker si moramo zapomniti v katero smer so se gibala vrata, da bi lahko šli v nasprotni smeri. Glede na to imamo stanji "ustavi gor", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala gor in stanje "ustavi dol", ki določa, da se bodo vrata ob naslednjem pritisku na gumb gibala dol. Če stanja ločimo tako, potem v stanju "ustavi gor" ostajamo toliko časa, dokler ne pritisnemo VRATA in jasno na motorju ni napake, se pravi kombinacija "10". Vrata se nato pomikajo gor (preidemo v stanje "vrata gor"). V tem stanju lahko tipko spustimo in vrata se pomikajo navzgor. To se dogaja toliko časa, dokler ne naletimo na pogoj ZAŠČITA='1' (se pravi kombinaciji

"11" in "01". Ko postane pogoj ZAŠČITA='1' se postavimo v stanje "ustavi dol" in v tem stanju ostajamo dokler vztraja pogoj ZAŠČITA='1' oz. dokler ne pritisnemo tipke VRATA='1' (kombinacija "10"). Takrat na podoben način preidemo v stanje "vrata dol", kjer ostanemo dokler ne naletimo na oviro (tla prostora recimo), ko preidemo v stanje "ustavi gor".

Takšna realizacija še zdaleč ni optimalna: Bolje bi bilo, če bi avtomat realizirali kot Mealy-ev tip. Dejanska realizacija ne vsebuje avtomata, temveč en T–FF in relejno logiko.

# RAZVOJ DIGITALNIH SISTEMOV

1. kolokvij  
14. 12. 2017

1. Izrazite podano logično funkcijo samo s Piercevimi operatorji. Morebitne negacije realizirajte s Piercevim operatorjem.

$$f(a, b, c) = ((a \cdot c) \downarrow \bar{b}) \oplus (a \uparrow c)$$

2. Določite redundance podane funkcije  $f$  tako, da bo nastala funkcija linearna, izračunajte koeficiente linearnosti ter funkcijo izrazite z uporabo izračunanih koeficientov.

$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5 - 7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

4. Pretvorite število  $8216_{10} = 2018_{16}$  v BCD zapis z uporabo "double dabble" algoritma.

## Rešitev 1. naloge

Realizacija s samimi Piercevimi (NOR, oziroma  $\downarrow$ ) operatorji zahteva pretvorbo funkcije v disjunktivno obliko (po možnosti normalno).

$$f(a, b, c) = ((a \cdot c) \downarrow \bar{b}) \oplus (a \uparrow c)$$

V ta namen moramo najprej operatorje (XOR, Pierce, Sheffer) v podani funkciji izpisati z disjunkcijami in konjunkcijami - začnemo na najvišjem nivoju, pri XOR funkciji in jo izpišemo. Obenem izpišemo še Shefferjev (NAND) operator. pri čemer izničimo nastalo dvojno negacijo zadnjega člena:

$$f(a, b, c) = ((a \cdot c) \downarrow \bar{b}) \cdot \overline{(a \cdot c)} + ((a \cdot c) \downarrow \bar{b}) \cdot (a \cdot c)$$

Iz dobljene oblike izpišemo še oba Pierceva (NOR) operatorja.

$$f(a, b, c) = ((a \cdot c) + \bar{b}) \cdot \overline{(a \cdot c)} + ((a \cdot c) + \bar{b}) \cdot (a \cdot c)$$

V prvem členu smo izničili dvojno negacijo, v drugem pa uporabimo de Morganov teorem nad predzadnjim členom in dobimo:

$$f(a, b, c) = ((a \cdot c) \cdot \overline{(a \cdot c)} + \bar{b} \cdot \overline{(a \cdot c)}) + ((a \cdot c) + \bar{b}) \cdot (a \cdot c)$$

Konjunkcija prvih dveh členov je nič, nad preostankom ponovno uporabimo de Morganov teorem, pri čemer izničimo nastalo dvojno negacijo nad členom  $b$ :

$$f(a, b, c) = \bar{b} \cdot (\bar{a} + \bar{c}) + \overline{(a \cdot c)} \cdot b \cdot (a \cdot c)$$

Iz izpisanega sledi, da je tudi zadnji člen enak 0, zato se izraz poenostavi v disjunktivno normalno obliko (DNO):

$$f(a, b, c) = \bar{b} \cdot (\bar{a} + \bar{c}) = \bar{a} \cdot \bar{b} + \bar{b} \cdot \bar{c}$$

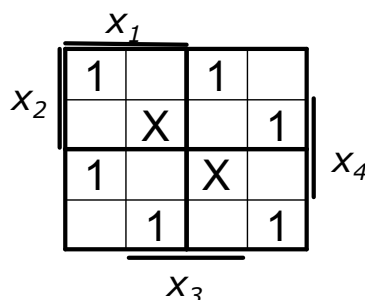
Nad členoma DNO ponovno uporabimo de Morganov teorem, da izrazimo prvi nivo s Piercevim operatorjem. V zgornjem izrazu nad vmesno konjukcijo izvedemo z dvojno negacijo:

$$f(a, b, c) = \bar{b} \cdot (\bar{a} + \bar{c}) = \overline{\overline{\bar{b} \cdot (\bar{a} + \bar{c})}} = b \downarrow (\bar{a} \downarrow \bar{c}) = b \downarrow ((a \downarrow a) \downarrow (c \downarrow c))$$



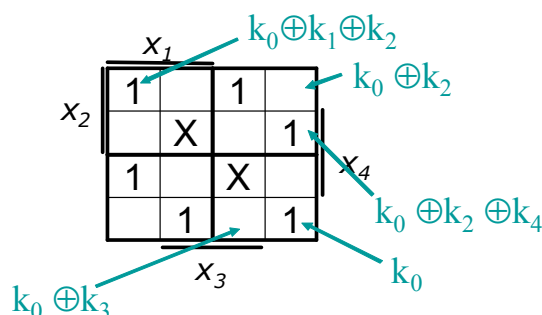
Rešitev 2. naloge:

Funkcijo najprej izrišemo v Veitchev diagram:



Če bo funkcija linearna, jo bomo lahko realizirali s pomočjo XOR funkcij. Linearnost funkcije ugotavljamo tako, da prepogibamo kvadrate diagrama: Začnemo v desnem spodnjem kotu (kjer je minterm 0) in prepognemo kvadrat navzgor, da se spremeni samo ena spremenljivka naenkrat ( $x_4$  postane 0 v prvi iteraciji).

Opazujemo, ali se prepogne na novi kvadrat čisto enako ali pa popolnoma negirano. Če postavimo obe redundanci na '1', lahko s prepogibanjem ugotovimo, da je funkcija linearna.



Podana funkcija je funkcija 4 spremenljivk, zato lahko njeno splošno izražavo kot linearno funkcijo pišemo kot:

$$f(x_1, x_2, x_3, x_4) = k_0 \oplus k_1 x_1 \oplus k_2 x_2 \oplus k_3 x_3 \oplus k_4 x_4$$

S pomočjo Veitch-evega diagrama izračunamo koeficiente.

Iz enačb sledi:  $k_0=1$  in  $k_0 \oplus k_3=0$ , kar pomeni  $1 \oplus k_3=0 \rightarrow k_3=1$ .

In če napišemo še enačbo za  $k_0 \oplus k_2=0$ , kar pomeni  $1 \oplus k_2=0$  sledi da je  $k_2=1$ .

Iz enačbe  $k_0 \oplus k_2 \oplus k_4=1$ , kar pomeni  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

Rezultati bodo objavljeni na domači strani predmeta.

### Rešitev 3. naloge:

Funkcija  $f$  je podana v obliki PKNO z redundancami.

$$f(x_1, x_2, x_3, x_4) = \&(1, 2, 5-7, 9, 10, 14) \text{ in } \&_x(0, 4, 11, 13)$$

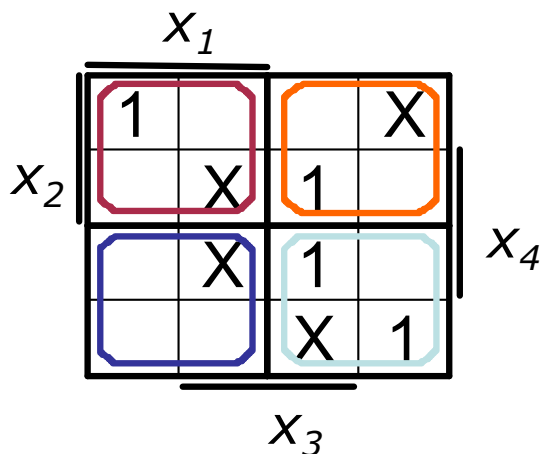
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f=0$  za vse maksterme in  $f=X$  za vse redundantne maksterme. Na preostala mesta vpišemo  $f=1$  in preberemo pri katerih mintermih je  $f=1$  oz.  $f=X$  ter funkcijo izrazimo v obliki PDNO.

$m$	$M$	$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	15	0	0	0	0	1
1	14	0	0	0	1	0
2	13	0	0	1	0	X
3	12	0	0	1	1	1
4	11	0	1	0	0	X
5	10	0	1	0	1	0
6	9	0	1	1	0	0
7	8	0	1	1	1	1
8	7	1	0	0	0	0
9	6	1	0	0	1	0
10	5	1	0	1	0	0
11	4	1	0	1	1	X
12	3	1	1	0	0	1
13	2	1	1	0	1	0
14	1	1	1	1	0	0
15	0	1	1	1	1	X

Dobimo:

$$f = V(0, 3, 7, 12) \text{ in } V_x(2, 4, 11, 15)$$

Dobljeno funkcijo vrišemo v Veitchev diagram. Ker iščemo najcenejšo realizacijo z izbiralnikom 4/1, bomo naredili razvoj po vseh kombinacijah naslovnih spremenljivk v Veitchevem diagramu. Če izberemo kot naslovni spremenljivki  $x_1$   $x_2$ , potem dobimo:



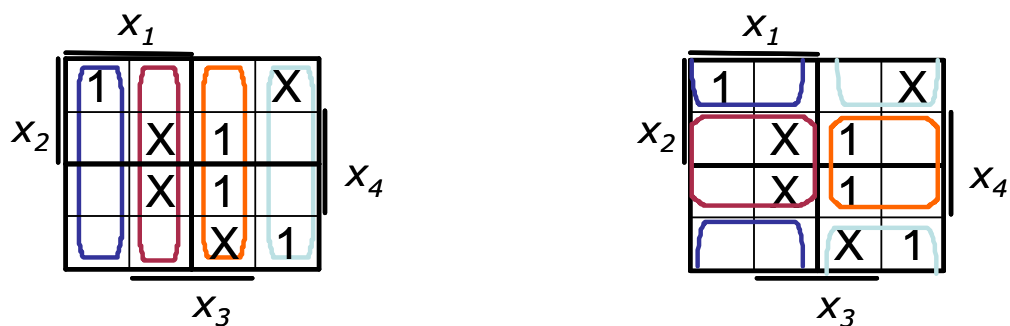
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete.

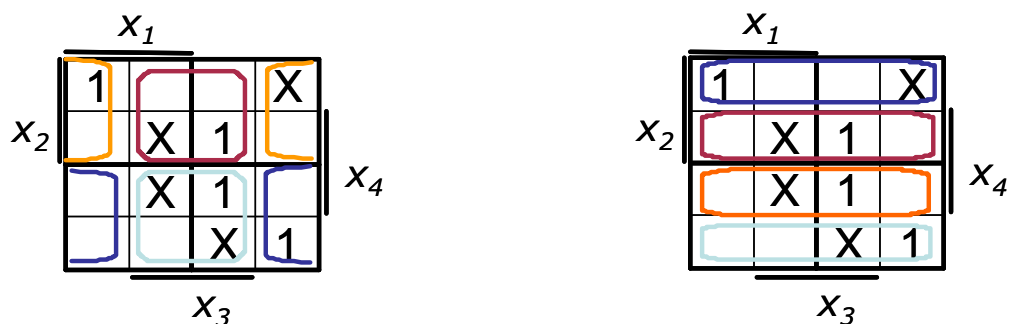
Rezultati bodo objavljeni na domači strani predmeta.

V zgornjem Veitchevem diagramu so označena vsa štiri polja funkcijskih ostankov, če izberemo vhodni spremenljivki  $x_1$   $x_2$ . Zgornji levi kvadrat (rdeč) pomeni, da bo to polje izbrano ko bosta  $x_1x_2="11"$ , oranžni kvadrat ko bo  $x_1x_2="01"$ , temno modri ko bo  $x_1x_2="10"$  in svetlo modri ko bo  $x_1x_2="00"$ . Vsakega od teh kvadratov poskušamo opisati s čimboli enostavno funkcijo: Vrednost zgornjega levega kvadrata je komplicirana, saj moramo vsako '1' opisati posebej: Za zgornjo '1' v tem kvadratu velja  $x_3x_4$ . Če bi v tem kvadratu postavili redundanco na '1' jo bi opisali kot  $x_3x_4$ . Funkcija bo torej  $x_3x_4 + x_3x_4$ , kar je enačba funkcije ekvivalence (XNOR). Podobno sklepanje velja za zgornji in spodnji desni kvadrat. Najbolj enostavna realizacija je spodnji levi kvadrat je konstanta '0', če postavimo redundanco na '0'. Za ostale možnosti realizacije moramo narisati še preostalih pet kombinacij dveh naslovnih vhodov.

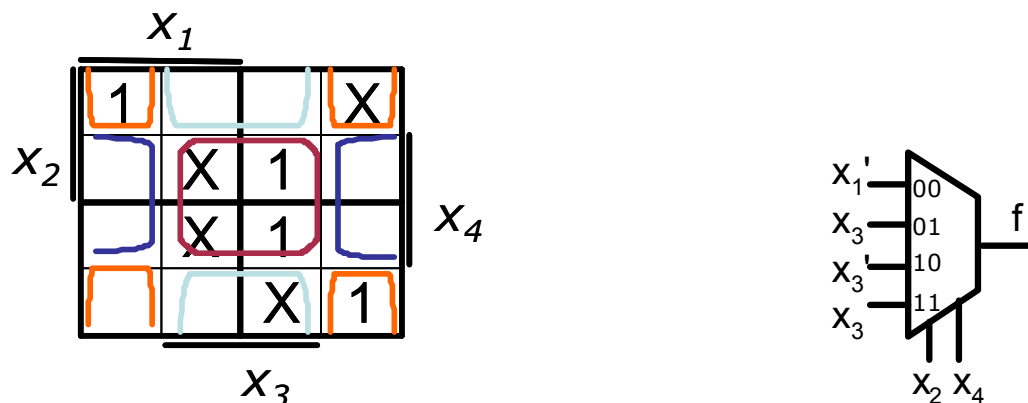
Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram, če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo inačice kvadratov, ki vsebujejo konstante (same '1' ali same '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo pri  $x_1x_3="10"$  najneugodnejšo funkcijo, saj vsebuje eno samo '1'. Pri razvoju po  $x_1$  in  $x_4$  nastopa ena sama '1' pri kombinaciji  $x_1x_4="10"$ .



Nato izberemo naslovni spremenljivki  $x_2$  in  $x_3$ , (levi diagram) in  $x_2$  in  $x_4$  (desni diagram). Pri razvoju po  $x_2$  in  $x_3$  imamo pri  $x_2x_3="00"$  najneugodnejšo funkcijo (moder), saj vsebuje eno samo '1'. Pri razvoju po  $x_2$  in  $x_4$  nikjer nimamo osamljene '1', zato se dajo funkcijski ostanki enostavno realizirati, če vse redundance postavimo na '1'.



Zadnja kombinacija naslovnih vhodov je  $x_3$  in  $x_4$ . Pri razvoju po  $x_3$  in  $x_4$  imamo pri  $x_3x_4="00"$  najneugodnejšo funkcijo (oranžen), saj dve '1' opišemo s funkcijo ekvivalence. Možno rešitev torej predstavlja kombinacija naslovnih vhodov  $x_2$  in  $x_4$ .



Vezje izbiralnika je v predlogah avditornih vaj na domači strani predmeta:

Logisim\MUX\mux\_4\_1\_f\_V\_0\_3\_7\_12\_in\_Vx\_2\_4\_11\_15.circ

Rešitev 4. naloge:

Število  $8216_{10} = 2018_{16} = 0010\ 0000\ 0001\ 1000_2$ .

Zapis posameznih števk v BCD zapisu se glasi: 1000 0010 0001 0110. Prve tri pomike smo v spodnji shemi preskočili, ker v tem primeru ne pride do prištevanja (100 ne povzroči prištevanja).

				1 0 0	0 0 0 0 0	0 0 0 1	1 0 0 0
			1 0 0	0 0 0 0 0 0 1 1 0 0 0			
			1 0 0 0	0 0 0 0 0 1 1 0 0 0			
			1 0 1 1	0 0 0 0 0 1 1 0 0 0			
		1	0 1 1 0	0 0 0 0 1 1 0 0 0			
		1	1 0 0 1	0 0 0 0 1 1 0 0 0			
		1 1	0 0 1 0	0 0 0 1 1 0 0 0			
		1 1 0	0 1 0 0	0 0 1 1 0 0 0			
		1 0 0 1	0 1 0 0	0 0 1 1 0 0 0			
	1	0 0 1 0	1 0 0 0	0 1 1 0 0 0			
	1	0 0 1 0	1 0 1 1	0 1 1 0 0 0			
	1 0	0 1 0 1	0 1 1 0	1 1 0 0 0			
	1 0	1 0 0 0	1 0 0 1	1 1 0 0 0			
	1 0 1	0 0 0 1	0 0 1 1	1 0 0 0			
	1 0 0 0	0 0 0 1	0 0 1 1	1 0 0 0			
1	0 0 0 0	0 0 1 0	0 1 1 1	0 0 0			
1	0 0 0 0	0 0 1 0	1 0 1 0	0 0 0			
1 0	0 0 0 0	0 1 0 1	0 1 0 0	0 0			
1 0	0 0 0 0	1 0 0 0	0 1 0 0	0 0			
1 0 0	0 0 0 1	0 0 0 0	1 0 0 0	0			
1 0 0	0 0 0 1	0 0 0 0	1 0 1 1	0			
1 0 0 0	0 0 1 0	0 0 0 1	0 1 1 0				

Srečno 2018 ☺ (šestnajstiško)!