

# RAZVOJ DIGITALNIH SISTEMOV

Izpit 11. 02. 2015

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

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

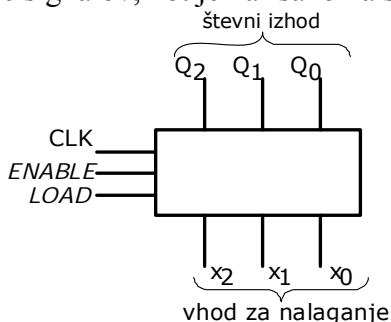
2. Realizirajte podano funkcijo  $f$  z redundancami s čim manj 4-bitnimi aritmetičnimi–logičnimi enotami (ALU). Negacije vhodnih spremenljivk izvedite z ALU.

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

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  $f$ .

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

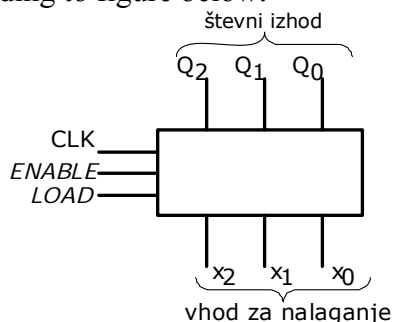
2. Implement a given function  $f$  using a single 4-bit arithmetic logic unit (ALU). Resulting negations of variables must be implemented within same ALU.

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

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:

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

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:

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

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

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

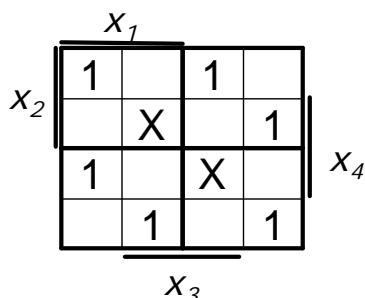
Funkcija v PKNO se torej glasi:

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

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

## 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  $1 \oplus 1 \oplus k_4=1 \rightarrow k_4=1$ .

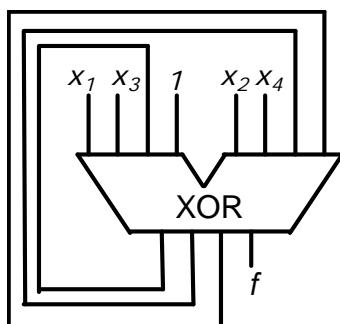
Analiziramo naprej in dobimo  $k_0 \oplus k_1 \oplus k_2=1$ , kar pomeni  $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$ .

Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

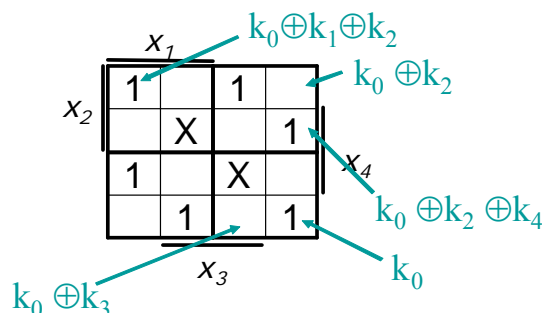
$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

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  $D_0 = Q_0'$

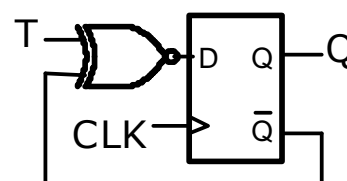
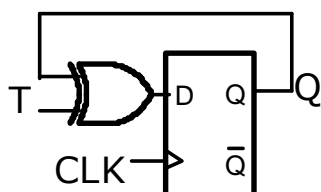
Za D<sub>1</sub> narišemo Veitchev diagram:

$D_1$ :

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

$$D_1 = Q_0 \oplus Q_1$$

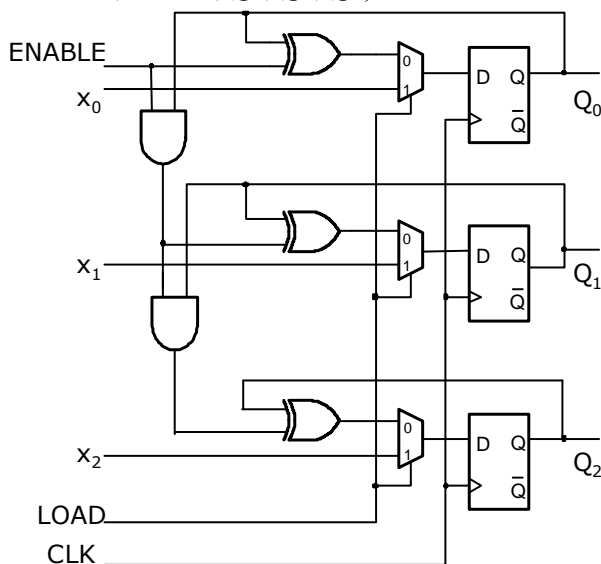
Če enačbo  $D_0 = Q_0'$  zapišemo kot  $D_0 = 1 \oplus Q_0$  in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:



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*,  $Q_2$ ,  $Q_1$ ,  $Q_0$ ).



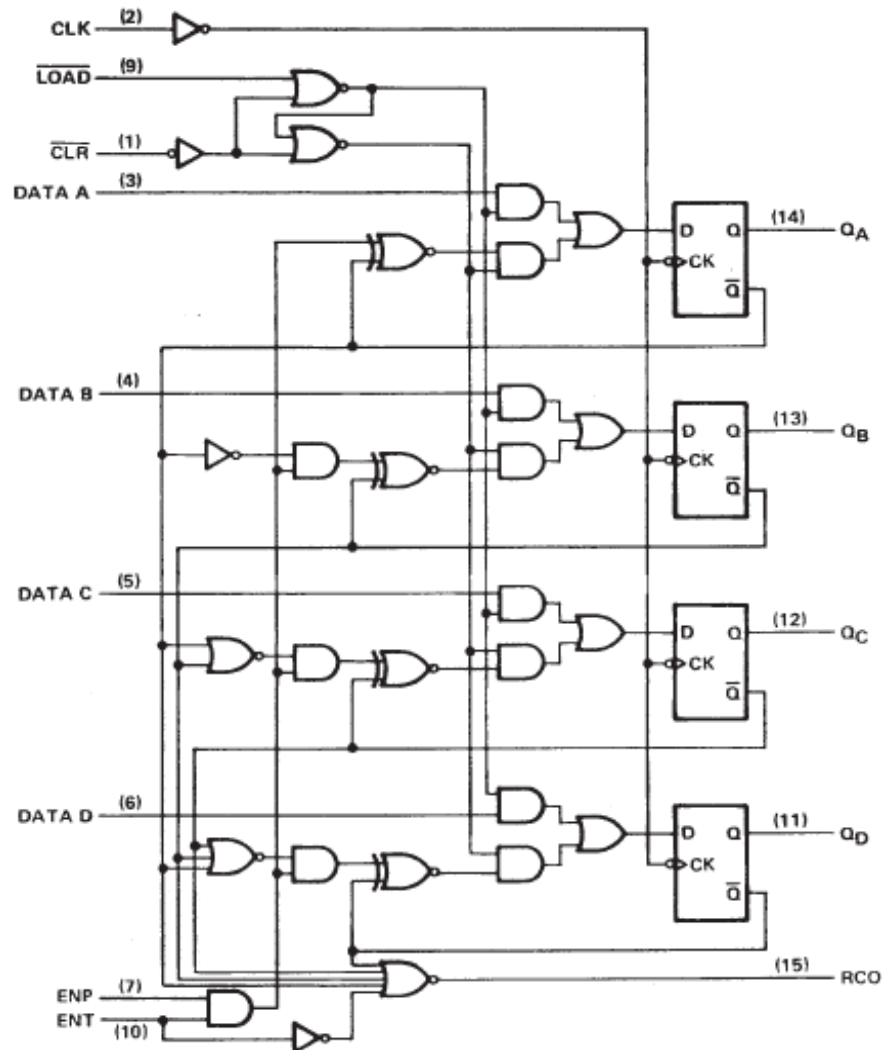
**Slika 2: Sinhroni števec z vzporednim nalaganjem (*LOAD*) in omogočanjem štetja (*ENABLE*) (3-bitna izvedba).**

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 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  $Q_D Q_C Q_B Q_A$ . 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:

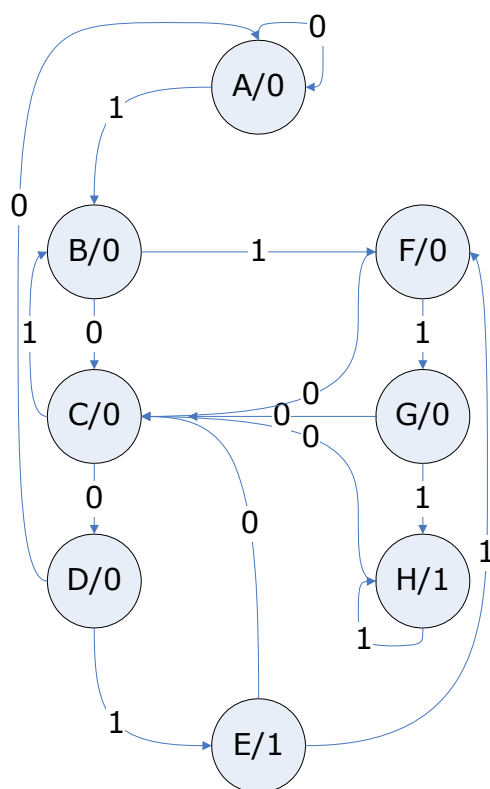


Diagram stanj začnemo risati tako, da najprej narišemo začetno stanje (A) v katerega se vračamo vedno, kadar sekvenca ne bo podobna tisti, ki jo zaznavamo (1111 oz. 1001). V tem stanju vztrajamo toliko časa, dokler je na vhodu '0', saj se nobena od sekvenc ne začneja z '0'. Ko pride na vhod prva '1', 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

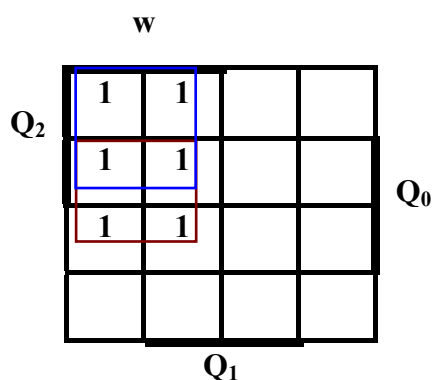
Za izvedbo bomo rabili najmanj 3 FF, saj je stanj 8. Izberemo zaporedno kodiranje stanj se pravi A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.

Vzbujalna tabela:

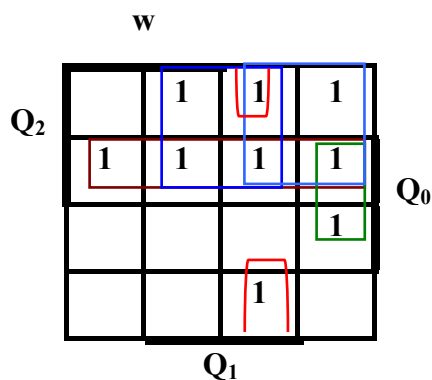
trenutno stanje				naslednje stanje						izhod
				w=0			w=1			
	Q <sub>2</sub> (t)	Q <sub>1</sub> (t)	Q <sub>0</sub> (t)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	Q <sub>2</sub> (t+1)	Q <sub>1</sub> (t+1)	Q <sub>0</sub> (t+1)	z
A	0	0	0	0	0	0	0	0	1	0
B	0	0	1	0	1	0	1	0	1	0
C	0	1	0	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

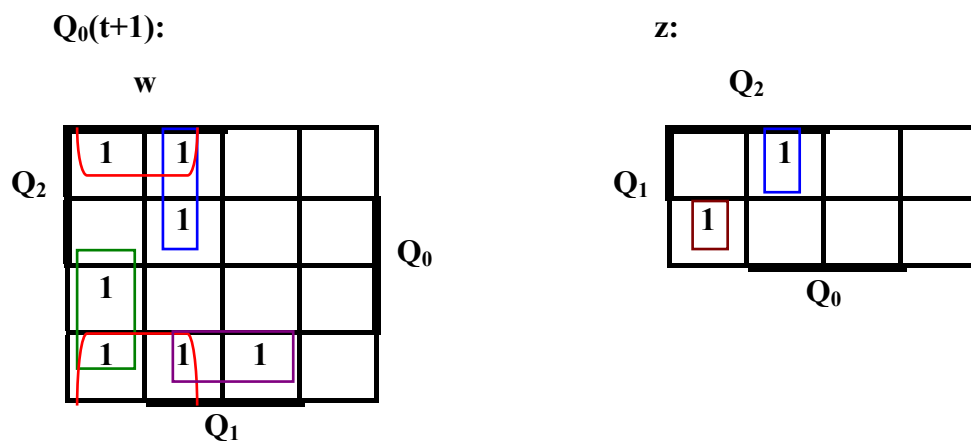
Iz vzbujalne tabele sestavimo tri Veitcheve diagrame:

$Q_2(t+1)$ :



$Q_1(t+1)$ :





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:

$$\begin{aligned} D_2 &= Q_2(t+1) = w \cdot Q_2(t) + \bar{w} \cdot Q_0(t) = w \cdot (Q_2(t) + Q_0(t)) \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + \bar{Q}_1(t) \cdot Q_0(t) + Q_1(t) \cdot \bar{Q}_0(t)) = \\ D_1 &= Q_1(t+1) = Q_2(t) \cdot (Q_1(t) + Q_0(t)) + \bar{w} \cdot (Q_2(t) + Q_1(t) \oplus Q_0(t)) \\ D_0 &= Q_0(t+1) = w \cdot \bar{Q}_0(t) + w \cdot Q_2(t) \cdot \bar{Q}_1(t) + w \cdot Q_2(t) \cdot Q_1(t) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t) = \\ D_0 &= Q_0(t+1) = w \cdot (\bar{Q}_0(t) + Q_2(t) \oplus Q_1(t)) + \bar{Q}_1(t) \cdot Q_1(t) \cdot \bar{Q}_0(t) \end{aligned} \quad (0.2)$$

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

