

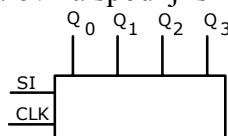
RAZVOJ DIGITALNIH SISTEMOV

Izpit 07. 02. 2020

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

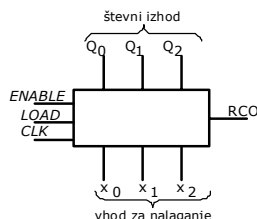
$$f^4 = \sum(0,1,6,7,14,15) \text{ in } \sum_x(3,5,9,11,13)$$

2. 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.



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 ₁	OP ₀	
0	0	motor stoji
0	1	motor pomika vrata navzgor
1	0	motor pomika vrata navzdol

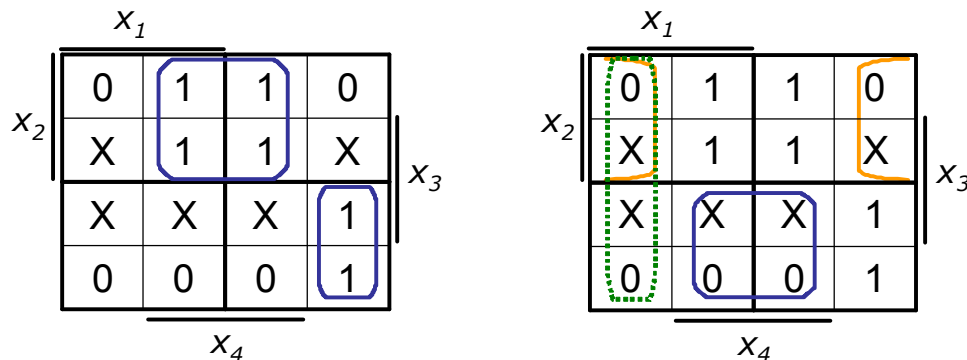
Če pritisnemo gumb VRATA, se vrata začno pomikati navzgor. Če na poti naletijo na oviro ali pridejo do zgornje končne lege, se motor ustavi. Če pritisnemo gumb VRATA ponovno, se začnejo gibati v smeri navzdol.

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.

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.

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

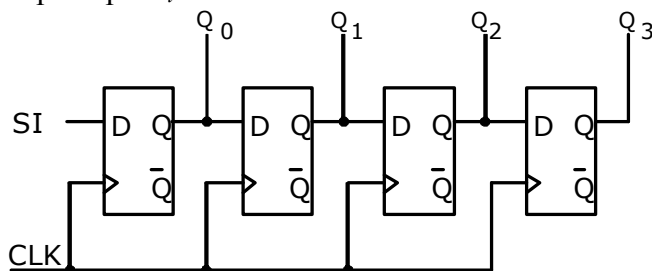
Za minimalno normalno obliko moramo določiti, katera izvedba funkcije je cenejša (manjši COST vezja).

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:

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 .



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

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:

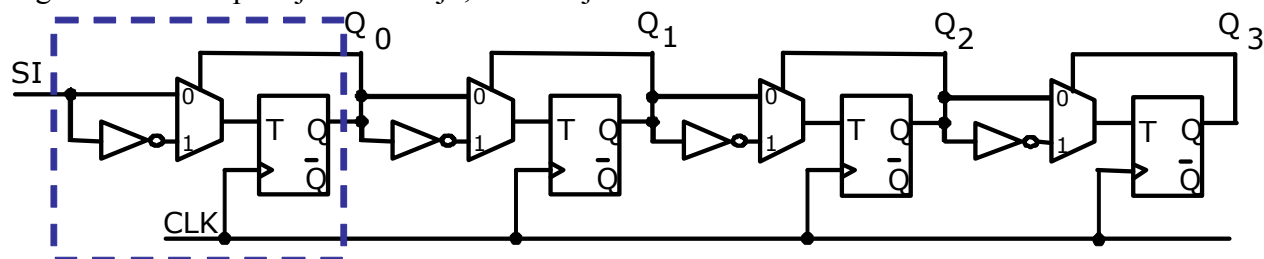
$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

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 realiziramo.

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.



Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

trenutno stanje			naslednje stanje			D-FF		
Q ₂	Q ₁	Q ₀	Q ₂	Q ₁	Q ₀	D ₂	D ₁	D ₀
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 D₀ se iz tabele vidi D₀ = Q₀'.

Za D₁ narišemo Veitchev diagram

D₁:

		Q ₂	
Q ₁	1	0	0
	0	1	1
		Q ₀	

$$D_1 = Q_0 \oplus Q_1$$

Podobno za D₂ narišemo Veitchev diagram

D₂:

		Q ₂	
Q ₁	1	0	1
	1	1	0
		Q ₀	

Za D₂ 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:

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

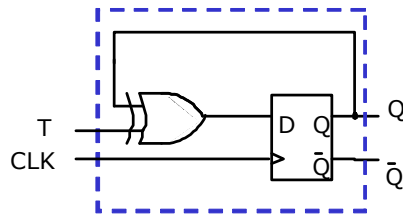
Upoštevamo definicijo XOR operacije (a ⊕ b = a' · b + a · b')

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

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".

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

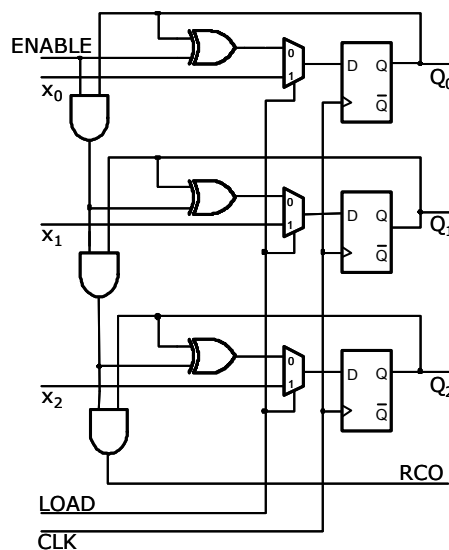
Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (ENABLE). Če vezje analiziramo, vidimo, da smo pravzaprav realizirali T–FF s pomočjo D–FF in XOR vrat, kot kaže spodnja slika:



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).

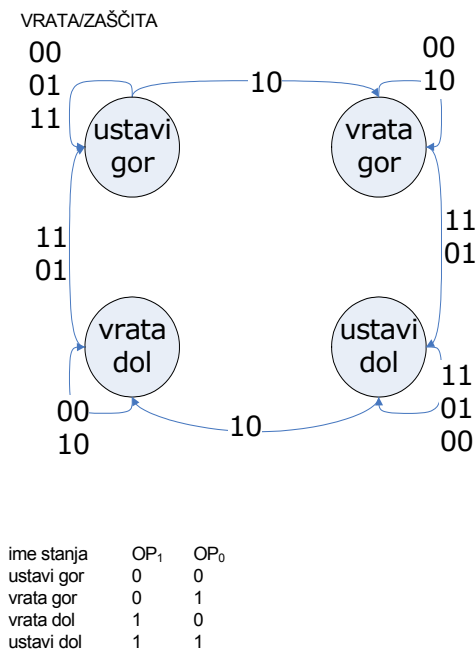
Če želimo z nastalim števcem šteti naraščajoče ... 2, 3, 4, 5, 2, 3, 4, 5 ..., moramo števec, ko le–ta pride do stanja 5 ($Q_2Q_1Q_0 = 101_2$) postaviti nazaj na stanje v stanje 2 ($Q_2Q_1Q_0 = x_2x_1x_0 = 010_2$), torej na LOAD vhod pripeljemo s pomočjo dodatnih dvovhodnih AND vrat.

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.

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.