

RAZVOJ DIGITALNIH SISTEMOV

Izpit

15. 06. 2020

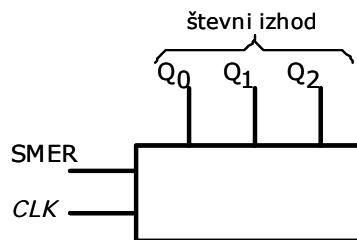
1. Realizirajte funkcijo f s čim manj izbiralniki 4/1.

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

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

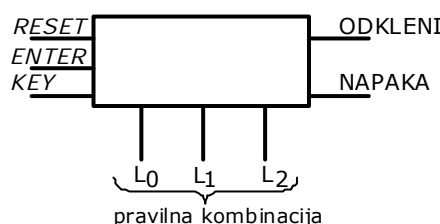
$$f(x_1, x_2, x_3, x_4) = V(0, 5, 6, 9, 10, 12) \text{ in } V_x(3, 15)$$

3. Prikažite sintezo 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. 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 ($KEY \equiv L_i$)
 - 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 .



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:

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

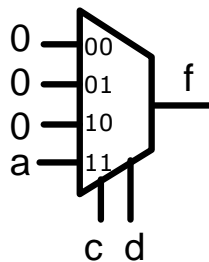
Dobimo dva člena in ju zapišemo v obliki PDNO, ki jo nato minimiziramo s pomočjo Veitch-evega diagrama ali z uporabo lastnosti združevanja Boole-ove algebre $x + \bar{x} = 1$:

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

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

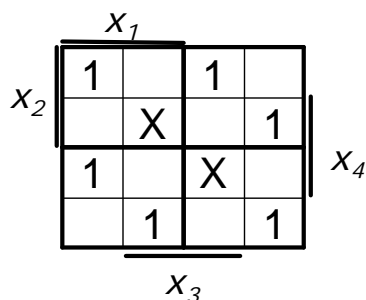
$$f_{MDNO}(a,b,c,d) = a \cdot c \cdot d \cdot (\bar{b} + b) = a \cdot c \cdot d$$

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).



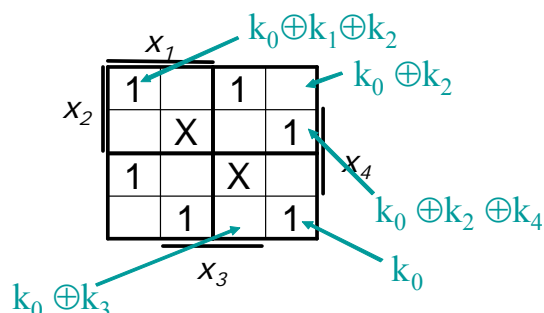
Rešitev 2. naloge:

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 $1 \oplus k_1 \oplus 1=0 \rightarrow k_1=1$.

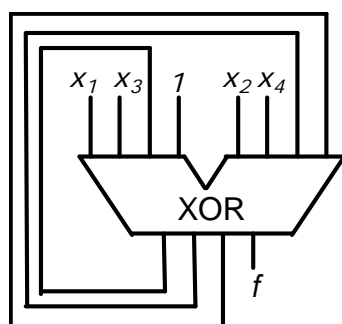
Vstavimo dobljene koeficiente v enačbo za splošno izražavo in dobimo:

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

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:



Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SMER	Q ₂	Q ₁	Q ₀	Q ₂	Q ₁	Q ₀	T ₂	T ₁	T ₀
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 T₀ se vidi, da je T₀='1'. Iz stolpca T₁ se vidi, da se ponavlja

vzorec 01, če je SMER='0' in 10, če je SMER='1'.

SMER	T ₁
0	Q ₀
1	Q ₀ '

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 T₂ se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

SMER			
Q ₂	1	0	0
	0	0	1
	0	0	1
	1	0	0
Q ₁			
Q ₀			

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

V enačbi za T₂ poiščemo podobnosti z enačbo za T₁: Enačba za T₁ vsebuje konjunkciji SMER·Q₀' in SMER'·Q₀, ki sta vsebovani tudi v enačbi za T₂, 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 74191¹. 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.

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

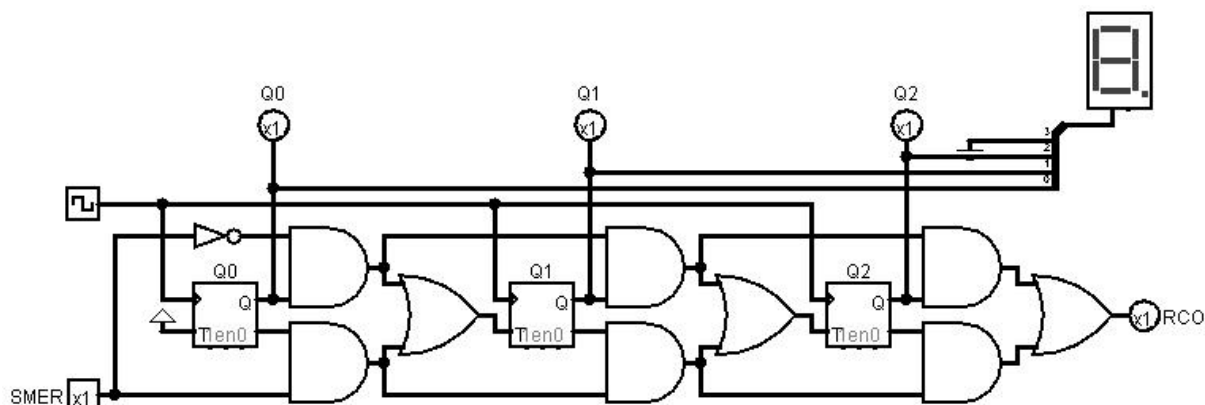
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. 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 74161².

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

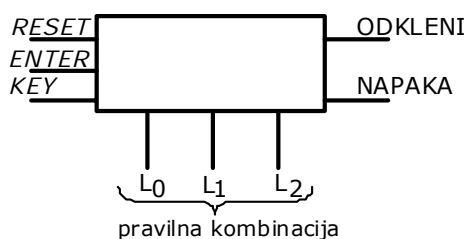
Rešitev 4. naloge:

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 ($KEY \equiv L_i$)
- 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 .



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 ($KEY \equiv L_0$) ali nepravilna ($KEY < > L_0$). 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*.

