

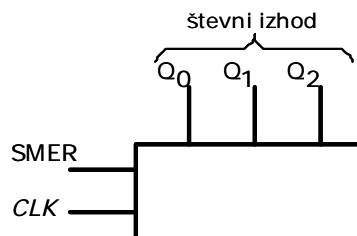
# RAZVOJ DIGITALNIH SISTEMOV

Izpit 04. 02. 2011

1. Z uporabo Boole-ovih postulatov preoblikujte podano funkcijo v minimalni normalni obliki tako, da bo primerna za realizacijo s čim manj dvovhodnimi vpoglednimi tabelami (look-up tabelami) v vezju FPGA.

$$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_1x_4x_6 + x_1x_5x_6 + x_2x_4x_6 + x_2x_5x_6 + x_3x_4x_6 + x_3x_5x_6 + x_7$$

2. Pretvorite število  $3AF_{16}$  v BCD zapis z uporabo "double dabble" algoritma.
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 za 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 **110** ali **101**, sicer je  $z='0'$ . Prekrivanje vzorcev je dovoljeno. Delovanje avtomata končnih stanj povzema spodnje časovno zaporedje vhoda in izhoda.

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

Rešitev 1. naloge:

Funkcijo v minimalni normalni obliki je treba preoblikovati tako, da bo primerna za realizacijo z dvovhodnimi vpoglednimi tabelami (look-up tabelami) v vezju FPGA.

$$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_1x_4x_6 + x_1x_5x_6 + x_2x_4x_6 + x_2x_5x_6 + x_3x_4x_6 + x_3x_5x_6 + x_7$$

V vseh konjunktivnih izrazih se nahaja spremenljivka  $x_6$ , tako da jo izpostavimo

$$f = x_6(x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_3x_4 + x_3x_5) + x_7$$

Nadalje poiščemo podobne spremenljivke, ki bi jih lahko izpostavili. Izpostavimo lahko  $x_4$  pri treh členih in  $x_5$  pri drugih treh členih.

$$f = x_6(x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_3x_4 + x_3x_5) + x_7$$

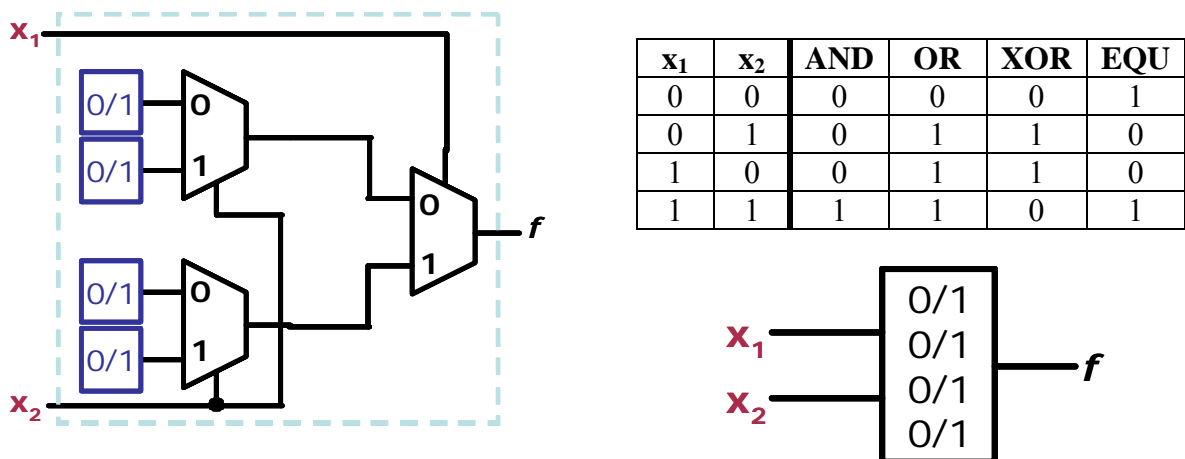
$$f = x_6(x_4(x_1 + x_2 + x_3) + x_5(x_1 + x_2 + x_3)) + x_7$$

$$f = x_6((x_1 + x_2 + x_3)(x_4 + x_5)) + x_7$$

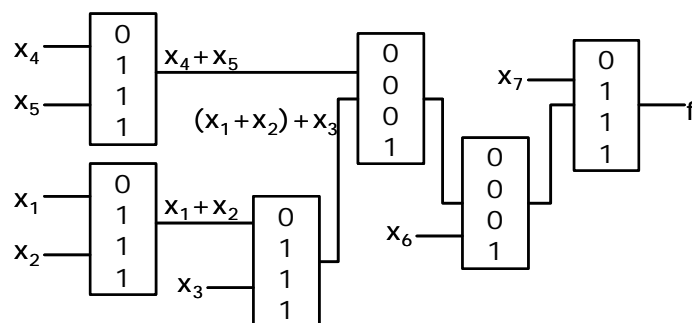
Nastaneta dva člena, ki jima je skupna disjunkcija  $x_1, x_2$  in  $x_3$ , ki jo ponovno lahko izpostavimo. Ker želimo funkcijo realizirati z dvovhodnimi vpoglednimi tabelami moramo še trovhodno disjunkcijo razstaviti v dva nivoja.

$$f = x_6(((x_1 + x_2) + x_3)(x_4 + x_5)) + x_7$$

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 (AND, OR, XOR, EQU). Tako lahko LUT uporabljamo za realizacijo funkcij v več nivojih.



[illegible]

### 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'**. Kot zanimivost omenimo dejstvo, ki se ponavlja pri večini sinhronih števec: Če namesto '1' na T<sub>0</sub> vodimo nek zunanji signal, nam to omogoča štetje števca (ENABLE oz. EN).

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:

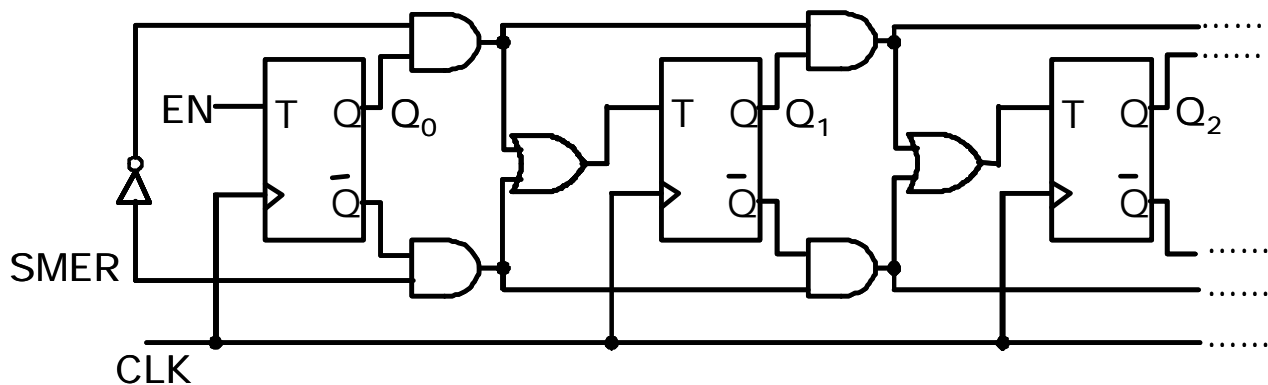
	SMER				
Q <sub>2</sub>	1	0	0	0	Q <sub>0</sub>
	0	0	1	0	
	0	0	1	0	
	1	0	0	0	
					Q <sub>1</sub>

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

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  $\overline{\text{SMER}} \cdot Q_0$ , 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 dvosmerne 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:

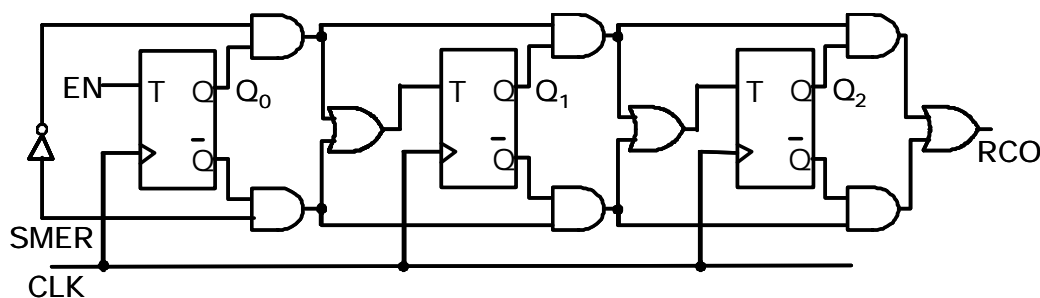
- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Signal EN je signal za omogočanje štetja – dokler je EN='0' se vsebina vseh T–FF ne spreminja, ampak ohranja trenutno stanje. Iz sheme števca je razviden tudi način razširitve števca na večje število bitov.

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:



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:

Naloga zahteva realizacijo z Moore-ovim 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 '0', tako da v tem stanju pomeni detekcijo sekvence "10X", v stanje D pa preidemo če se pojavi na vhodu '1', kar pomeni detekcijo sekvence "11X". Od tod naprej se pomaknemo v stanje E, če se v stanju C pojavi na vhodu logična '1', oz. če se v stanju D pojavi na vhodu logična '0'. V stanju E je izhod  $z=1$ , saj smo zaznali sekvenco.

Neuspela zaznavanja: Če se v stanju D pojavi '1' na vhodu, potem gre za sekvenco "111" na vhodu – kar še vedno pomeni, da preidemo nazaj v stanje B, saj je prekrivanje vzorcev dovoljeno. Drugače se diagram obnaša, ko smo v stanju C 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'. Podobno velja za stanje E.

