

# RAZVOJ DIGITALNIH SISTEMOV

Izpit

12. 12. 2013

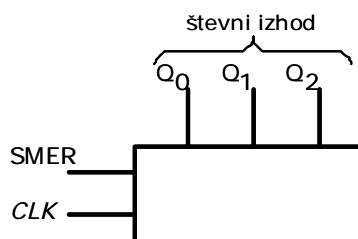
1. Realizirajte podano funkcijo  $f$  z redundantnimi makstermi z enim izbiralnikom 4/1.

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

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.

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

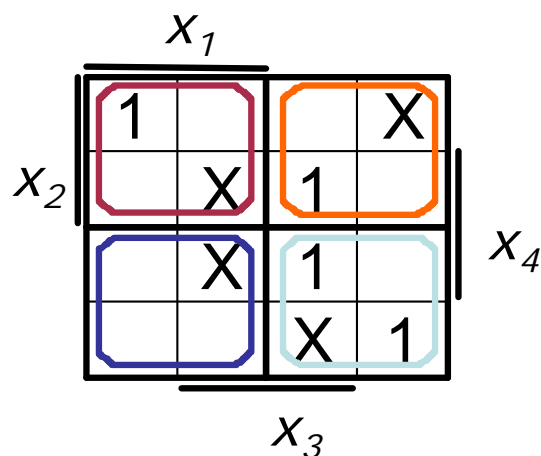
Za potrebe realizacije jo najprej pretvorimo v obliko PDNO. To storimo tako, da maksterme preslikamo v minterme. V pravilnostno tabelo funkcije najprej zapišemo številke mintermov ( $m$ ) in pripadajoče številke makstermov ( $M$ ). Vpišemo  $f='0'$  za vse maksterme in  $f='X'$  za vse redundantne maksterme. 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.

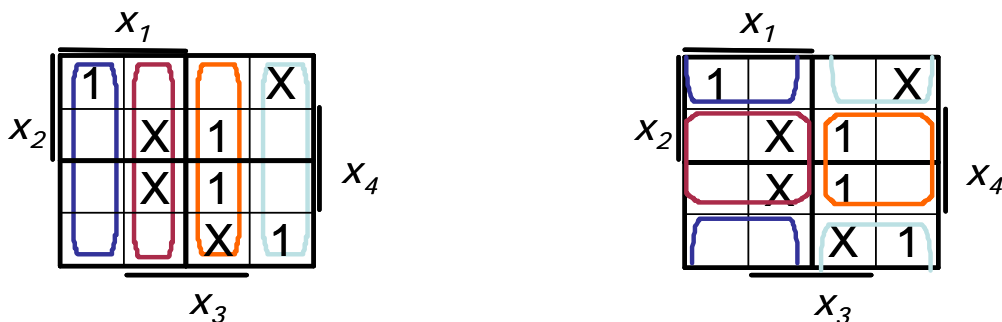
Če izberemo kot naslovni spremenljivki  $x_1$  in  $x_3$ , dobimo levi Veitchev diagram,

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

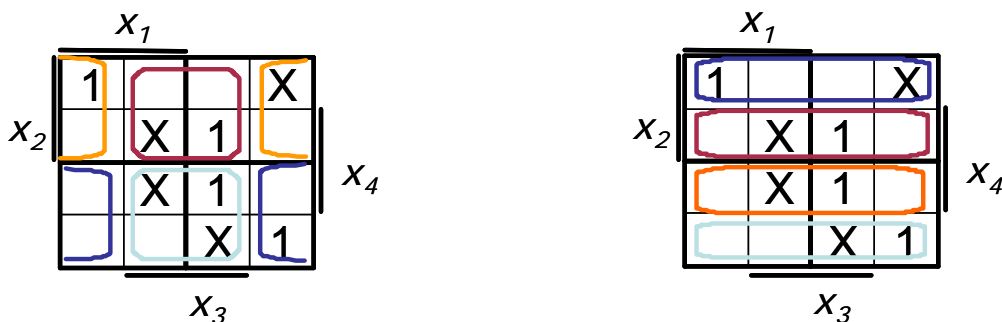
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete

Rezultati bodo objavljeni na: <https://studij.fe.uni-lj.si>

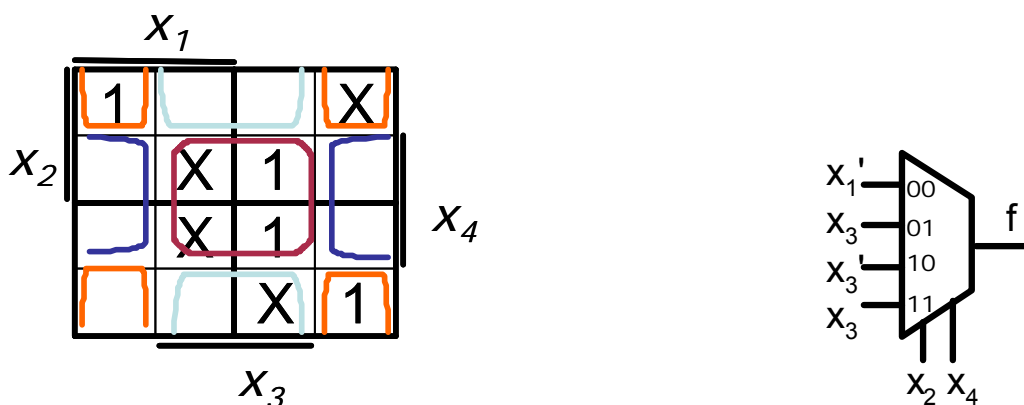
če  $x_1$  in  $x_4$ , pa desnega. Podobno kot v prejšnjem primeru poiščemo realizacije ustreznih kvadratov in iščemo najenostavnejšo realizacijo: Izogibamo se veliko različnim funkcijam in iščemo drugače kvadratov, ki vsebujejo konstante (samo '1' ali samo '0'). Pri razvoju po  $x_1$  in  $x_3$  imamo 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.

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

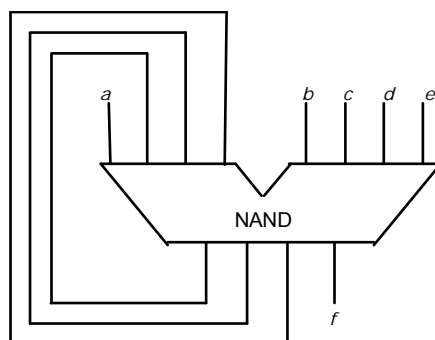
Podobno storimo še enkrat:

$$f(a, b, c, d, e) = \overline{\overline{\left( \overline{\overline{(a \cdot b) \cdot c}} \right) \cdot d \cdot e}}$$

Dobljene NAND operatorje predstavimo z oklepaji.

$$f(a, b, c, d, e) = \left( \left( (a \uparrow b) \uparrow c \right) \uparrow d \right) \uparrow e$$

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

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>

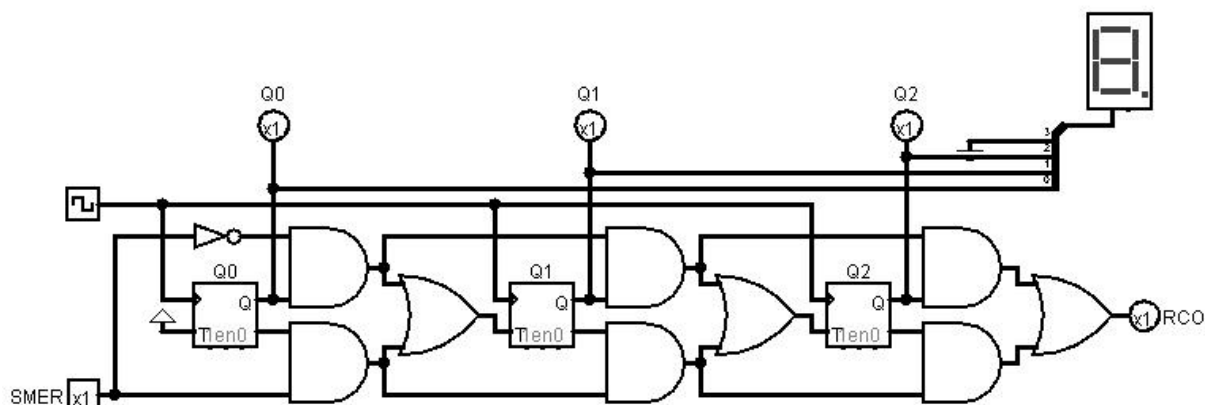
Ko enkrat narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. 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 števecv tako, da izdelane 3 bitne števecve 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

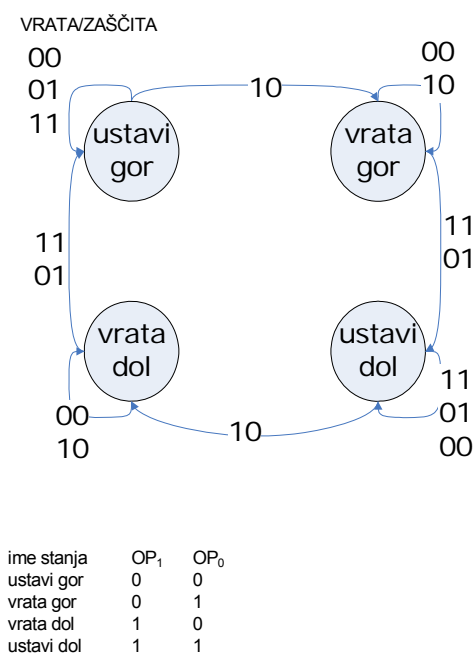
Večina števecv je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števecv:

- desetiški (BCD) števeci, 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 števecv 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".