UNIVERZA V MARIBORU

Fakulteta za elektrotehniko,

računalništvo in informatiko

Bogdan DUGONIK

DIGITALNI SISTEMI

UNIVERZA V MARIBORU

Fakulteta za elektrotehniko, računalništvo in informatiko

Bogdan DUGONIK

DIGITALNI SISTEMI

MARIBOR, November 2002

DIGITALNI SISTEMI SKRIPTA

Avtor: mag. Bogdan Dugonik, univ. dipl. inž. el.

Uredil in oblikoval: mag. Bogdan Dugonik, univ. dipl. inž. el.

Strokovna recenzija: doc. dr. Tatjana Kapus, univ. dipl. inž. el.

izr. prof. dr. Zmago Brezočnik, univ. dipl. inž. el.

Lektor: doc. dr. Tatjana Kapus, univ. dipl. inž. el.

Vrsta publikacije: Skripta

Izdala: Fakulteta za elektrotehniko, računalništvo in informatiko

Tisk: Založniško tiskarska dejavnost Tehniških fakultet v Mariboru

Naklada: 200 izvodov

Drugi ponatis 2002

CIP - Kataložni zapis o publikaciji

Univerzitetna knjižnica Maribor

681.32(075.8)

DUGONIK, Bogdan

Digitalni sistemi : [skripta] / Bogdan Dugonik. – Maribor : Fakulteta za elektrotehniko, računalništvo in informatiko, 2000

ISBN 86-435-0314-2

COBISS-ID 44779009

KAZALO

KAZALO SLIK

KAZALO TABEL

1 UVOD

Sodobni sistemi za procesiranje, shranjevanje in prenos informacij so digitalni, kar pomeni, da procesirajo, shranjujejo in prenašajo informacije v digitalni obliki. Digitalni sistemi so natančnejši, zanesljivejši in bolj fleksibilni od analognih sistemov. Glavni predstavnik digitalnih sistemov je digitalni računalnik, ki zaznamuje informacijsko dobo, v kateri živimo.

Digitalni sistemi so za študente elektronike in telekomunikacij uvodni predmet s področja digitalne tehnike. Pri predavanjih se spoznajo z osnovnimi gradniki in metodami snovanja, ki se uporabljajo pri gradnji kombinacijskih in sinhronih sekvenčnih digitalnih sistemov. Obravnava se paleta različnih predstavitev digitalnih sistemov: pravilnostna tabela, stikalni diagram, logična vrata, časovni diagram, tranzistorsko vezje, tabela prehajanja stanj, diagram prehajanja stanj, blokovni diagram itd. Poleg nekaterih obveznih bolj teoretičnih klasičnih poglavij hitro predstavimo sodobna načrtovalska orodja in tehnike strukturiranega načrtovanja.

Vaje pri predmetu Digitalni sistemi so sestavljene iz treh delov: avditornih vaj, laboratorijskih in računalniških vaj. Pri avditornih vajah se teorija digitalnih sistemov preslika v praktične primere, laboratorijske vaje pa zagotavljajo utrjevanje konceptov, ki jih študent spozna na predavanjih in avditornih vajah. Za dobrega načrtovalca so poleg solidnega teoretičnega znanja potrebne tudi praktične izkušnje pri sestavljanju digitalnih sistemov, ki pa si jih lahko pridobi le z intenzivnim delom v laboratoriju. Laboratorijske vaje potekajo tako, da **študent vso teoretično pripravo opravi že doma**. V laboratoriju nato na posebni laboratorijski plošči sestavi in preizkusi že zasnovan digitalni sestav. Obsežnost izbranih sestavov je prilagojena majhnemu številu razpoložljivih ur za laboratorijske vaje. Pri računalniških vajah se študent spozna z osnovami sodobnega načina snovanja in realizacije digitalnih sistemov. Osnova pri računalniških vajah je program LogicWorks, zato je le korak tudi do izvedbe malo zahtevnejših projektov, ki jih izvajamo deloma v laboratoriju, deloma doma.

Na področju digitalnih sistemov lahko danes najdemo veliko literature, vendar je ta v glavnem izdana v angleškem jeziku. Pri pisanju tega dela sem se oprl predvsem na učni načrt pri predmetu DIGITALNI SISTEMI.

Po uvodu sledijo kratka navodila za opravljanje laboratorijskih vaj. V tretjem delu so podane laboratorijske vaje, na katere se študent pripravi že doma, kasneje pa jih samostojno izvede v laboratoriju. V četrtem delu je nekaj napotkov za opravljanje računalniških vaj ter navodila za izdelavo projekta. V petem delu je izčrpen opis funkcij programskega paketa LogicWorks, sledi pa mu šesti del, v katerem je podanih trinajst projektnih nalog. Vsak študent si izbere eno izmed njih. Realizira jo pri računalniških vajah in doma. V sedmem delu sem podal nekatere rešitve nalog iz šestega dela. Rešitve naj bi študentu rabile predvsem kot vodilo pri reševanju njegovega projekta ter kot pomoč za povečanje njegove ustvarjalnosti. V osmem delu najdemo opise nekaterih pri laboratorijskih in računalniških vajah najpogosteje uporabljenih elementov, v zadnjem, devetem delu pa je seznam uporabljene literature.

2 NAVODILA ZA OPRAVLJANJE LABORATORIJSKIH VAJ

Pri delu v laboratoriju naj vsak udeleženec vaj upošteva naslednja pravila:

- Laboratorijske vaje se opravijo predvidoma v treh delih.
- Na vsakem delovnem mestu je po en študent.
- Prisotnost na laboratorijskih vajah je obvezna.
- Urnik vaj s seznamom za prijavo k vajam se razpiše v tednu pred pričetkom cikla laboratorijskih vaj.
- Za vsako vajo, ki jo bo študent opravljal v laboratoriju, mora imeti izdelano pripravo iz pričujoče skripte. Pri tem naj upošteva zahteve, ki so za vsako vajo podane v odstavku **Priprava na vajo**. Študent, ki bo prišel na vaje brez priprav, bo moral laboratorij zapustiti.
- Seznaniti se mora z laboratorijsko ploščo, ki je prikazana na sliki 2.2-1 (podnožje za integrirana vezja, priključne puše, stikala, indikatorji logičnih signalov itd.).
- Upoštevati mora vse varnostne zahteve za delo v laboratoriju.
- Poznati in upoštevati mora vse tehnične podatke za elemente, ki jih bo pri delu uporabljal. Tehnični podatki so zbrani v prilogi na koncu skripte.
- Vsako opravljeno vajo da v pregled asistentu.
- Uspešno opravljene laboratorijske vaje so pogoj za pristop k izpitu.

2.1 NAPOTKI ZA DELO Z INTEGRIRANIMI VEZJI IZ DRUŽIN TTL IN CMOS

Družina TTL:

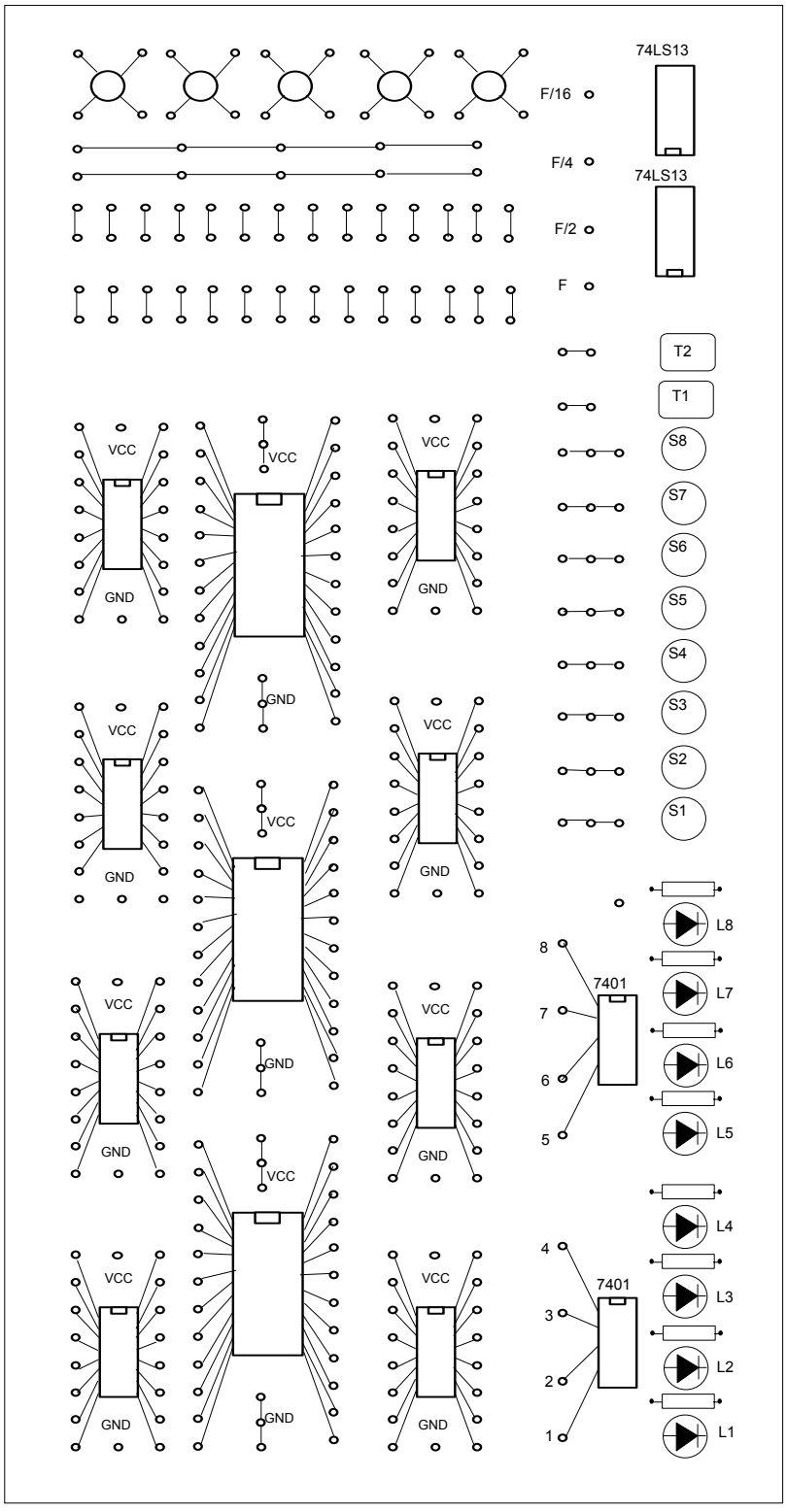
- Napajalna napetost U_{CC} = 5 V $\pm$ 0,25 V.
- Maksimalni vhodni napetostni nivo U_{vhmax} < 5,5 V.
- Na izhod preklopnega elementa ne vsiljujte zunanjega napetostnega (logičnega) nivoja.
- Nepovezan ("odprt") vhod se v TTL logiki interpretira kot visok logični nivo.

Družina CMOS:

- Napajalna napetost U_{DD} = 3 V do 18 V (zaradi TTL elementov, ki so na laboratorijski plošči, uporabimo le 5 V).
- Maksimalni vhodni napetostni nivo $U_{vhmax} = U_{DD} + 0,5$ V.
- Vsi neuporabljeni vhodi preklopnih elementov integriranega vezja morajo imeti določene vhodne nivoje (logična stanja 0 ali 1).

2.2 KRATEK OPIS LABORATORIJSKE PLOŠČE

Iz laboratorijske mize pripeljemo na laboratorijsko ploščo (slika 2.2-1) napajalno napetost 5 V. Na laboratorijski plošči so vse napajalne točke V_{CC} oz. GND med seboj povezane, tako da lahko posameznim integriranim vezjem pripeljemo napajalno napetost iz najbližjih napajalnih puš GND in V_{CC}. Zaradi preglednosti na sliki 2.2-1 nismo narisali vseh medsebojnih povezav puš GND oz. V_{CC}. Na laboratorijski plošči je za namestitev čipov razporejenih enajst podnožij tipa DIL. Osem podnožij je takih, da lahko vanje namestimo čipe z največ šestnajstimi nožicami, tri podnožja pa dopuščajo namestitev čipov z največ štiriindvajsetimi nožicami. Podnožja za čipe na laboratorijski plošči imajo za vsako nožico čipa svojo priključno pušo. Kadar vstavimo v podnožje čip s številom nožic, manjšim od velikosti podnožja, moramo biti pri povezovanju še posebej previdni. Vse tri puše nad stikali S1 do S8 so povezane. Njihov napetostni (logični) nivo je odvisen od položaja stikala. Signal za urine impulze dobimo bodisi iz delilnika frekvence (puše z oznakami F do F/16) bodisi iz ročnih impulznikov (tipki T1 in T2). Vpliv impulznih konic zaradi odskakovanja mehanskih kontaktov tipk T1 in T2 je v plošči odpravljen. Logične vrednosti izhodov opazujemo na svetlečih diodah L1 do L8. Čipa 74 LS 01 sta namenjena ojačevanju logičnih signalov za svetleče diode. Na laboratorijski plošči imamo tudi puše za vstavljanje pasivnih elementov (uporov in kondenzatorjev) in podnožja, v katera vstavljamo tranzistorje.

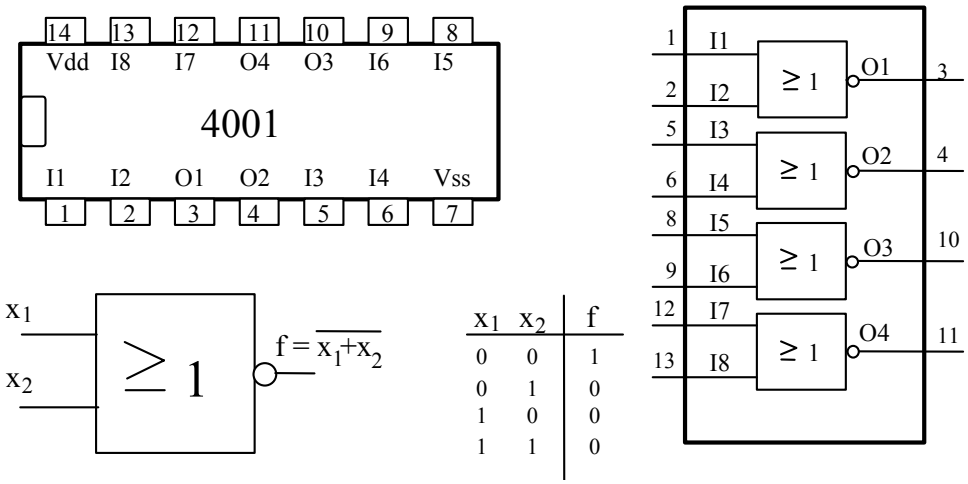


Slika 2.2-1. Laboratorijska plošča.

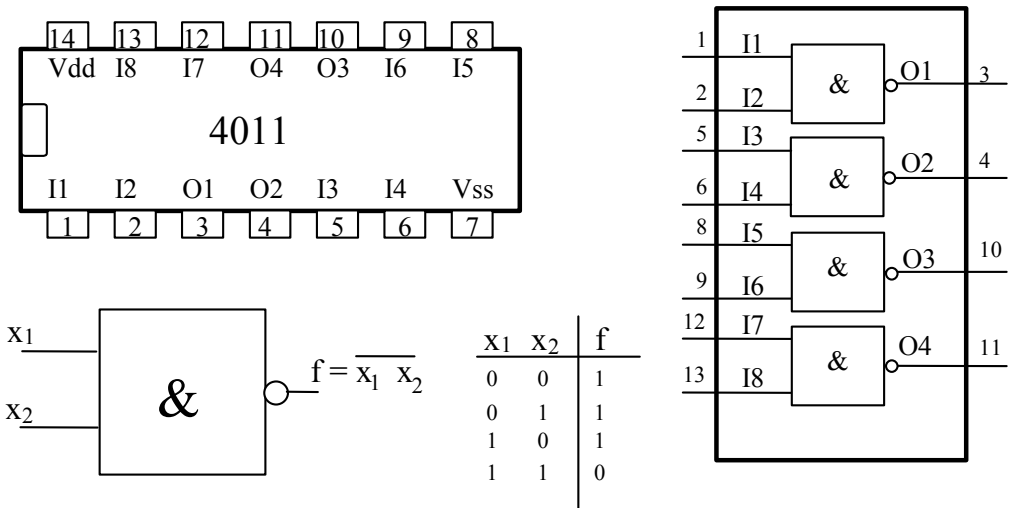
3 LABORATORIJSKE VAJE

3.1 Pretvorba funkcijskega sistema

Besedilo naloge: S Shefferjevimi in Pierceovimi logičnimi vrati sestavite in preizkusite logična vezja za negacijo, konjunkcijo in disjunkcijo. Sestavite in preizkusite tudi logično vezje, ki bo opravljalo funkcijo dvovhodnih logičnih vrat XOR, tako da boste uporabili eno samo integrirano vezje CD4011 (štiri Shefferjeva vrata).



Slika 3.1-1. Dvovhodna Pierceova vrata (NOR) in razporeditev v čipu CD4001.



Slika 3.1-2. Dvovhodna Shefferjeva vrata (NAND) in razporeditev v čipu CD4011.

Priprava na vajo: Doma opravite naloge iz točk 2, 3 in 4. Pomagajte si z načrti za integrirana vezja, ki so zbrani v prilogi priročnika. Na vseh shemah morate oštevilčiti priključke elementov.

1. V integriranih vezjih CD4001 in CD4011 preverite delovanje logičnih vrat. V integriranem vezju CD4001 (CD4011) so štiri dvovhodna vrata NOR (NAND). Razporeditev priključkov vrat, preklopna funkcija in pravilnostna tabela so podani na sliki 3.1-1 (sliki 3.1-2). Vsaka vrata testirajte tako, da jim prek stikal S1 - S8 na laboratorijski plošči pripeljete na vhode logične vrednosti 0 oz.1, izhod vrat pa vežete na svetleče diode L1 - L8. Opazujte logične vrednosti na izhodu vrat in jih primerjajte z vrednostmi iz njihove pravilnostne tabele. Če ugotovite, da vrata v čipu ne delujejo pravilno, ga zamenjajte z drugim.

2. Zapišite enačbe in narišite vezalne sheme za realizacijo osnovnih logičnih funkcij (NOT, AND, OR) z logičnimi vrati NAND.

3. Zapišite enačbe in narišite vezalne sheme za realizacijo osnovnih logičnih funkcij (NOT, AND, OR) z logičnimi vrati NOR.

4. Funkcijo XOR želimo realizirati z enim integriranim vezjem CD4011 (s štirimi dvovhodnimi vrati NAND). Funkcijo najprej pretvorimo v naslednjo nenormalno obliko:

$f(x_1, x_2) = x_1\overline{x}_2 + \overline{x}_1 x_2 = x_1(\overline{x}_1 + \overline{x}_2) + x_2(\overline{x}_1 + \overline{x}_2)$

Zapišite zgornjo funkcijo s Shefferjevim funkcijsko polnim sistemom:

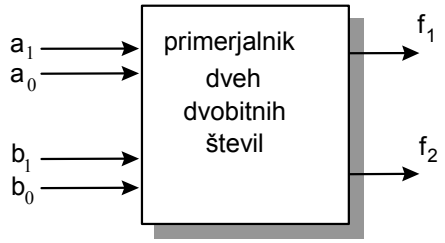
$f(x_1, x_2) =$

Narišite vezalno shemo za vrata XOR z enim integriranim vezjem CD4011.

5. Na laboratorijski plošči sestavite in testirajte funkcije iz točk 2, 3 in 4.

3.2 Primerjalnik

Besedilo naloge: Realizirajte preklopno vezje za primerjanje dveh dvobitnih števil, A = a_1a_0 in B = b_1b_0 (slika 3.2-1) Vezje ima izhoda f_1 in f_2. Odvisnost funkcijskih vrednosti izhodov f_1 in f_2 od števil A in B je podana v tabeli 3.2-1.



Slika 3.2-1. Blokovna shema primerjalnika.

Tabela 3.2-1. Kode za izhode primerjalnika.

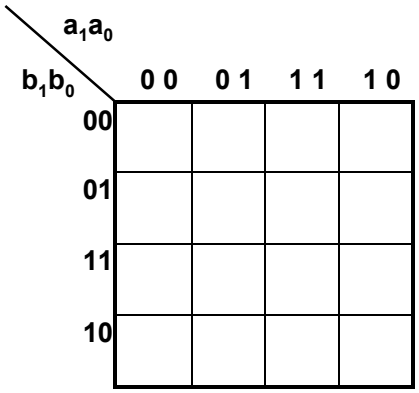
	f_1	f_2
A<B	1	1
A=B	1	0
A>B	0	1

Priprava na vajo: Doma realizirajte naloge iz točk 1, 2 in 3.

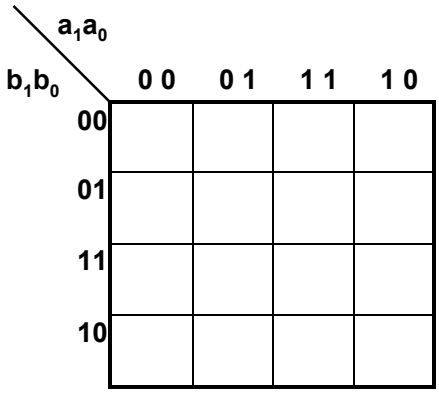
1. Izpolnite pravilnostno tabelo in za funkciji f_1 in f_2 narišite Karnaughov diagram. Poiščite MDNO in MKNO funkcij f_1 in f_2.

1	0	1	0	f_1	f_2
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

f_1:



f_2:



f_1 =

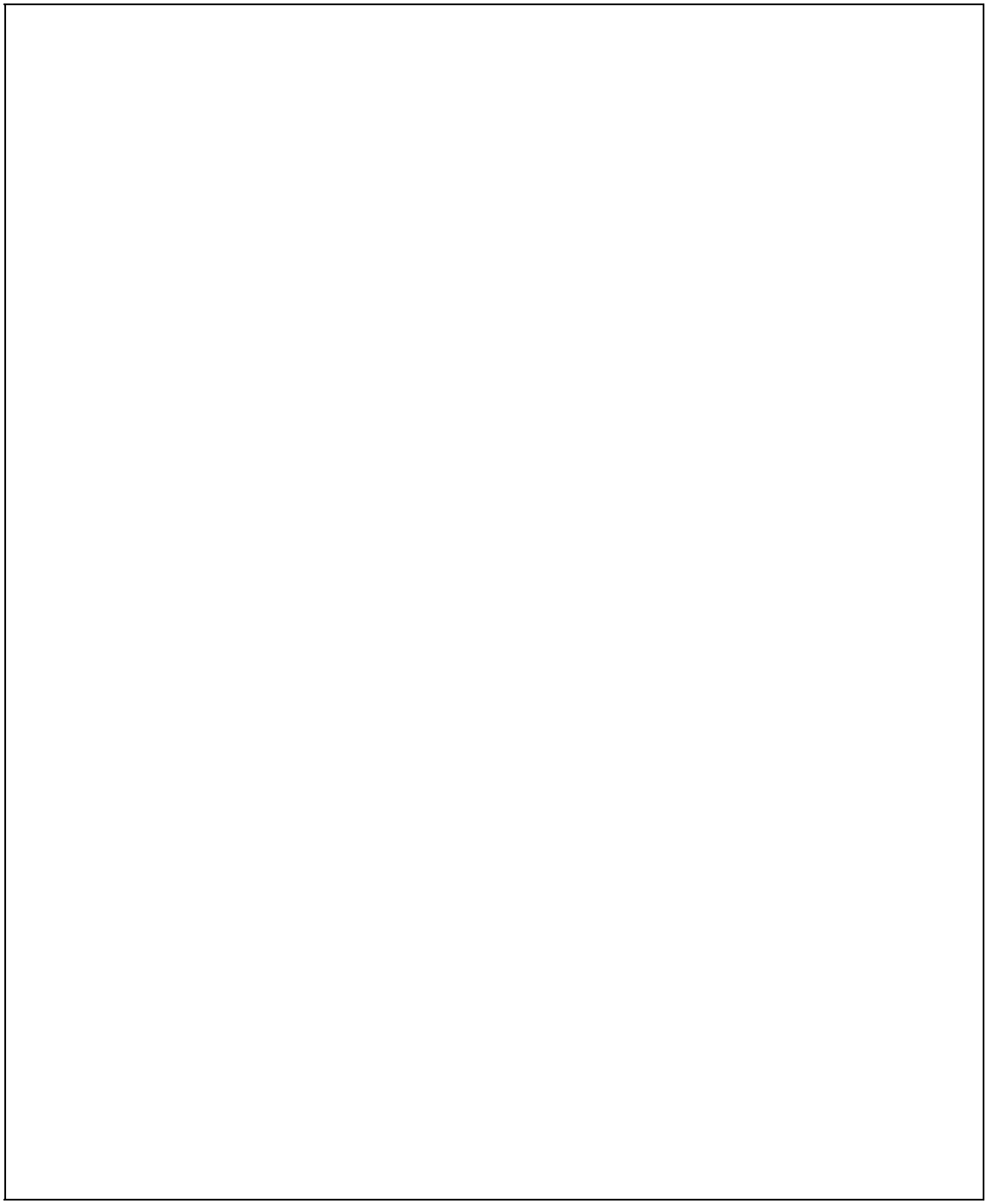
f_2 =

2. Preklopni funkciji f_1 in f_2 zapišite samo s Shefferjevim ali samo s Pierceovim funkcijsko polnim sistemom. Izberite tiste operatorje, s katerimi boste lahko realizirali funkciji z manjšim številom logičnih vrat.

f_1 =

f_2 =

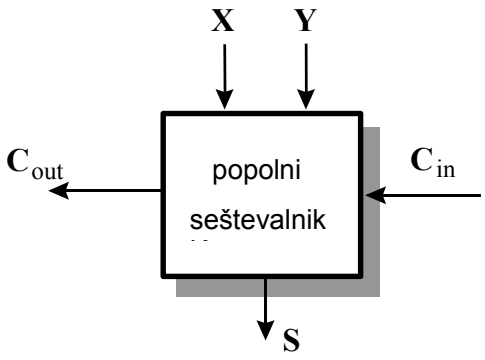
3. Narišite vezalno shemo primerjalnika in oštevilčite priključke čipov.



4. Vezje iz točke 3 realizirajte na laboratorijski plošči in preverite njegovo delovanje glede na tabelo 3.2-1.

3.3 Popolni seštevalnik z logičnimi vrati

Besedilo naloge: Realizirajte popolni seštevalnik (slika 3.3-1). Uporabite minimalno število Shefferjevih ali Pierceovih logičnih vrat, ki jih izberite iz družine standardnih integriranih vezij CMOS (glejte prilogo).



Slika 3.3-1. Blokovna shema popolnega seštevalnika.

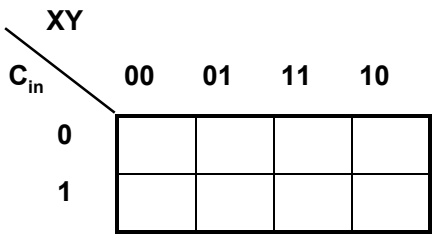
Priprava na vajo: Ugotovite, kakšne so razlike v delovanju polovičnega in popolnega seštevalnika. Doma opravite naloge iz točk 1, 2, 3, 4 in 5.

1. Izpolnite pravilnostno tabelo, ki ponazarja delovanje popolnega seštevalnika. S predstavlja funkcijo za vsoto, C_{out} pa funkcijo za izhodni prenos.

X	Y	C_{in}	S	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

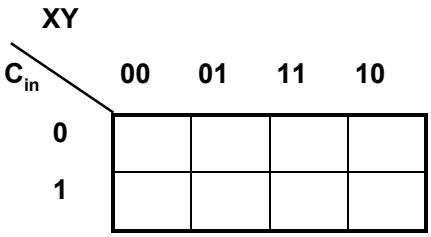
2. Vstavite vrednosti iz pravilnostne tabele v Karnaughov diagram in minimizirajte funkciji za S in C_{out}.

S:



Minimizirana funkcija $S(X, Y, C_{in}) =$

C_{out}:



Minimizirana funkcija $C_{out}(X, Y, C_{in}) =$

3. Pretvorite minimizirani funkciji S in C_{out} v Shefferjev ali Pierceov funkcijsko poln sistem:

$S(X, Y, C_{in}) =$

$C_{out}(X, Y, C_{in}) =$

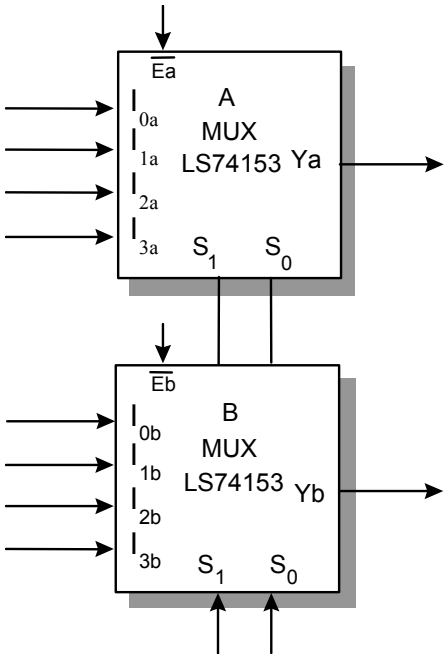
4. Narišite shemo popolnega seštevalnika z razpoložljivimi Shefferjevimi ali Pierceovimi logičnimi vrati (glejte prilogo). Vsi priključki čipov morajo biti obvezno oštevilčeni.

5. Narišite, kako bi s popolnimi seštevalniki realizirali večbitni seštevalnik, npr. štiribitnega.

6. Sestavite vezje popolnega seštevalnika iz točke 4 na laboratorijski plošči in preizkusite pravilnost njegovega delovanja glede na pravilnostno tabelo iz točke 1.

3.4 Popolni seštevalnik z multipleksorjem

Besedilo naloge: Realizirajte popolni seštevalnik z uporabo dvojnega multipleksorja LS74153 iz družine integriranih vezij tipa TTL (slika 3.4-1). Dvojni multipleksor 74LS153 ima v enem integriranem vezju dva enaka dela - multipleksor A in multipleksor B. Multipleksor A (B) ima podatkovne vhode I_{0a}, I_{1a}, I_{2a} in I_{3a} (I_{0b}, I_{1b}, I_{2b} in I_{3b}) in izhod Y_a (Y_b). Naslovna vhoda S_1 in S_0 sta skupna za oba dela. Podrobnejši podatki o delovanju vezja 74LS153 so zbrani v prilogi. Z multipleksorjem A realizirajte funkcijo za vsoto S, z multipleksorjem B pa funkcijo za prenos C_{out}. Na naslovna vhoda (S_0, S_1) pripeljite spremenljivki X in Y, na podatkovne vhode (I_{ia} oz. I_{ib}, $0 \leq i \leq 3$) pa funkcijske ostanke $S(C_{in})$ in $C_{out}(C_{in})$.



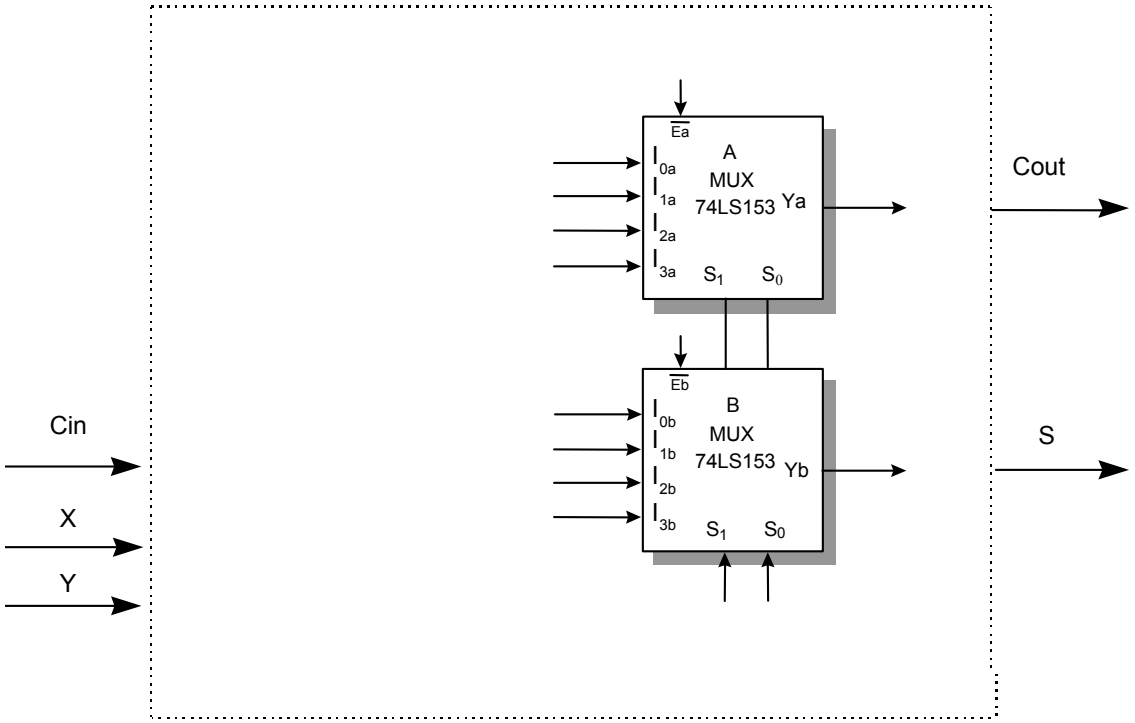
Slika 3.4-1. Blokovna shema dvojnega 4-vhodnega multipleksorja 74LS153.

Priprava na vajo: Doma opravite naloge iz točk 1 in 2. V prilogi preglejte, kako deluje podani multipleksor.

1. Izpolnite pravilnostno tabelo za popolni seštevalnik in zapišite funkcijske ostanke $S(C_{in})_i$ oz. $C_{out}(C_{in})_i$, ki jih moramo priključiti na vhode I_{ia} oz. I_{ib} , $0 \le i \le 3$.

X	Y	C_{in}	S	C_{out}	i	$S(C_{in})_i$	$C_{out}(C_{in})_i$
0	0	0			0		
0	0	1					
0	1	0			1		
0	1	1					
1	0	0			2		
1	0	1					
1	1	0			3		
1	1	1					

2. Narišite shemo za popolni seštevalnik z uporabo multipleksorja 74LS153.



3. Na laboratorijski plošči sestavite in preizkusite vezje iz točke 2.

3.5 Pretvornik kodov

Besedilo naloge: Sestavite preklopno vezje, ki pretvarja kod 8421-BCD v kod Excess-3. Vezje realizirajte z uporabo klasičnih Shefferjevih ali Pierceovih logičnih vrat. Izberite tisti tip vrat, ki bo po minimizaciji dal vezje z manjšim številom čipov. Pri minimizaciji upoštevajte tudi nedoločeno vrednost X.

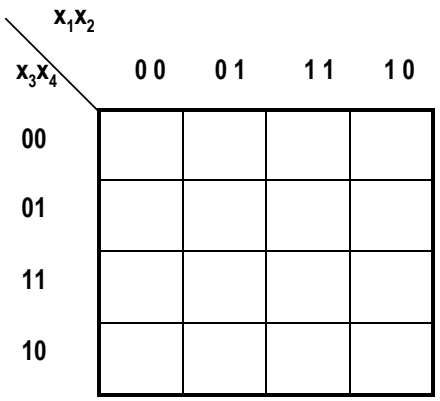
Priprava na vajo: Doma opravite naloge iz točk 1, 2, 3 in 4. Dopolnite pravilnostno tabelo s kodom Excess 3. (Kodo Excess 3 za številko i, $0 \leq i \leq 9$, dobimo tako, da njeni 8421-BCD kodi prištejemo 3 v dvojiški obliki.)

1. Pravilnostna tabela s kodom Excess-3.

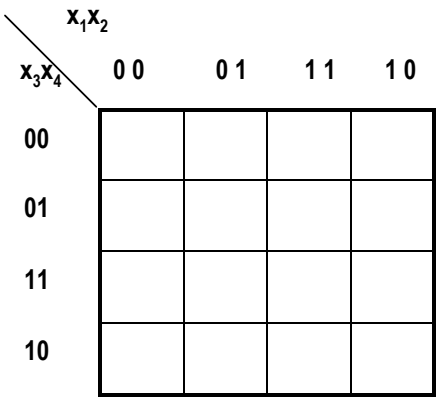
Kod 8421-BCD					Kod Excess-3			
i	x_1	x_2	x_3	x_4	z_1	z_2	z_3	z_4
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

2. Minimizirajte preklopne funkcije $z_i(x_1, x_2, x_3, x_4)$, $1 \leq i \leq 4$, s pomočjo Karnaughovih diagramov. Funkcije zapišite v MDNO ali MKNO, tako da jih boste lahko realizirali z minimalnim številom logičnih vrat (Pierce, Sheffer). Pri minimizaciji upoštevajte tudi nedoločene vrednosti X.

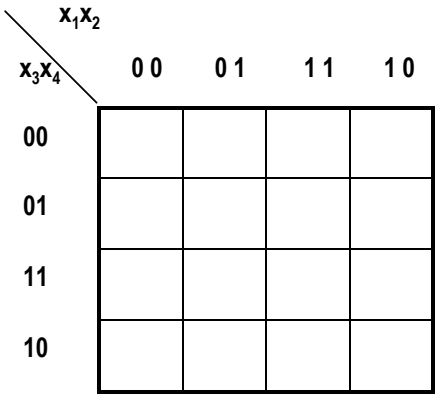
z_1:



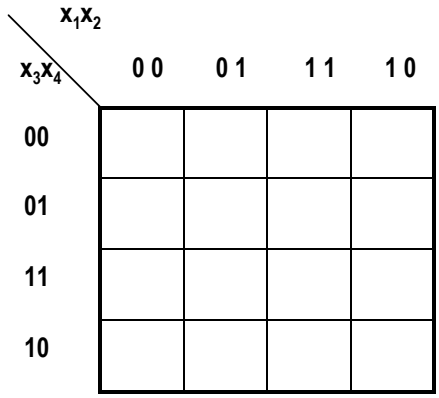
z_2:



z_3:



z_4:



MDNO ali MKNO funkcij $z_i(x_1, x_2, x_3, x_4)$, $1 \leq i \leq 4$, so:

$z_1(x_1, x_2, x_3, x_4) =$

$z_2(x_1, x_2, x_3, x_4) =$

$z_3(x_1, x_2, x_3, x_4) =$

$z_4(x_1, x_2, x_3, x_4) =$

3. Minimalne oblike preklopnih funkcij pretvorite v Shefferjev ali Pierceov funkcijsko poln sistem:

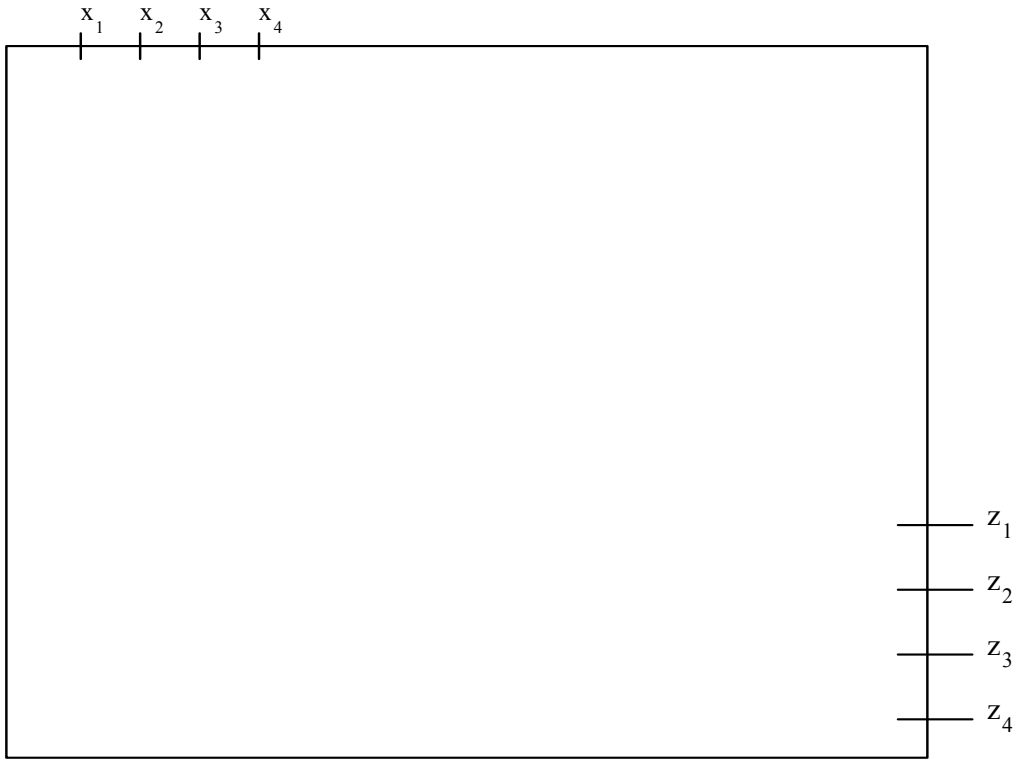
$z_1(x_1, x_2, x_3, x_4) =$

$z_2(x_1, x_2, x_3, x_4) =$

$z_3(x_1, x_2, x_3, x_4) =$

$z_4(x_1, x_2, x_3, x_4) =$

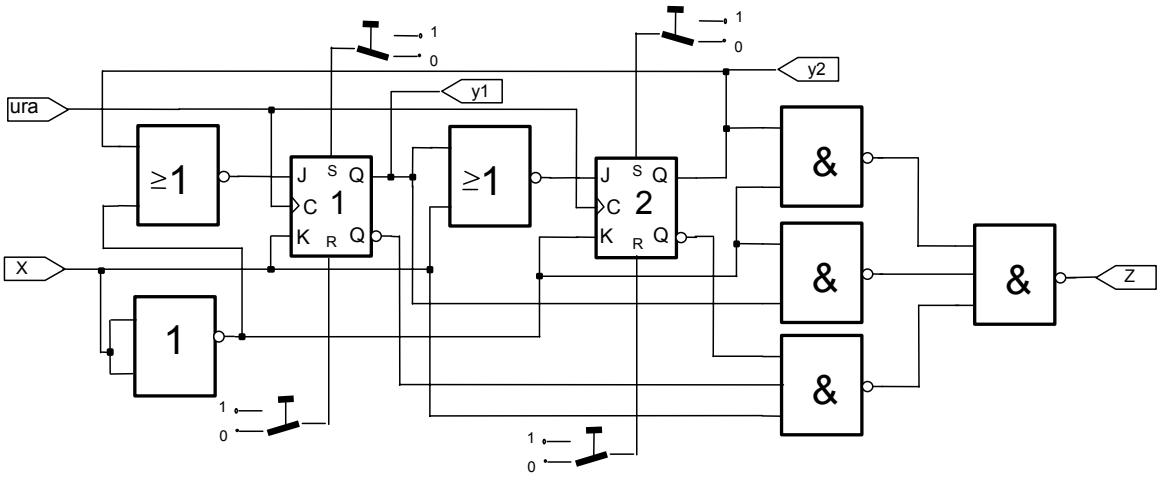
4. Narišite preklopno vezje glede na zapisane funkcije v točki 3.



5. Realizirajte vezje iz točke 4 na laboratorijski plošči in preizkusite pravilnost njegovega delovanja glede na pravilnostno tabelo v točki 1.

3.6 Analiza sekvenčnega vezja

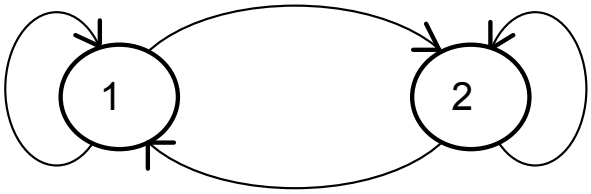
Besedilo naloge: Analizirajte sinhrono sekvenčno vezje s slike 3.6-1. Zanj narišite diagram prehajanja stanj in sestavite tabelo prehajanja stanj. Primerjajte rezultate teoretične analize z izmerjenimi rezultati, ki jih dobite pri analizi fizično realiziranega vezja po shemi. Pred povezovanjem na laboratorijski plošči oštevilčite vse priključke elementov.



Slika 3.6-1. Shema sinhronega sekvenčnega vezja.

Priprava na vajo: Preštudirajte pomnilne enačbe in obnašanje pomnilnih celic tipa D, T, JK in RS. Oglejte si podatke o pomnilnih celicah v prilogi. Doma opravite naloge iz točk 1 in 2.

1. Izpolnite diagram prehajanja stanj in narišite tabelo prehajanja stanj za pomnilno celico JK.



2. Izvedite teoretično analizo vezja.

Izhod Q prve pomnilne celice označite s spremenljivko stanja y_1, izhod Q druge pomnilne celice pa s spremenljivko stanja y_2.

- Zapišite krmilne enačbe pomnilnih celic.

$J_1 =$

$K_1 =$

$J_2 =$

$K_2 =$

- Zapišite enačbo za naslednjo vrednost vsake spremenljivke stanja.

$y_1^+ =$

$y_2^+ =$

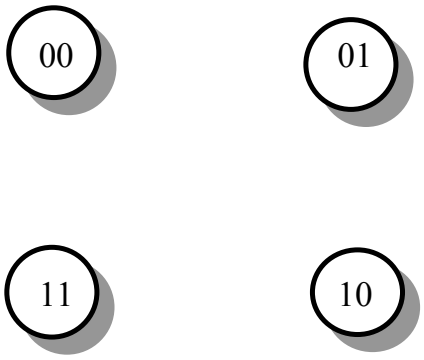
- Zapišite izhodno enačbo.

$z =$

- Izpolnite tabelo prehajanja stanj.

sedanje stanje		naslednje stanje /izhod x =	
		0	1
y_1	y_2	$y_1^+ y_2^+/z$	$y_1^+ y_2^+/z$
0	0	/	/
0	1	/	/
1	0	/	/
1	1	/	/

- Narišite diagram prehajanja stanj. Notranje stanje vezja je označeno z y_1y_2.



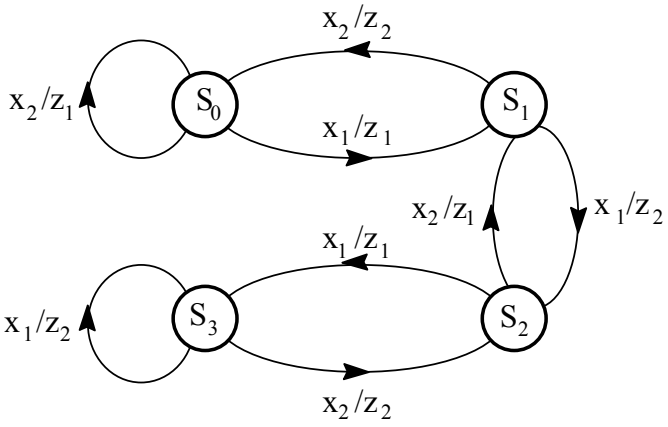
3. Izvedite laboratorijsko analizo vezja.

- Sestavite vezje s slike 3.6-1 na laboratorijski plošči. Vpišite izmerjene rezultate analize v tabelo in jih primerjajte z rezultati, ki ste jih dobili pri teoretični analizi vezja v točki 2.

;edanje stanje		naslednje stanje / izhod x =	
		0	1
y_1	y_2	$y_1^+ y_2^+/z$	$y_1^+ y_2^+/z$
0	0	/	/
0	1	/	/
1	0	/	/
1	1	/	/

3.7 Sinteza sinhronega sekvenčnega vezja

Besedilo naloge: Izvedite sintezo sinhronega sekvenčnega vezja z minimalnim številom logičnih vrat in pomnilnih celic po podanem diagramu prehajanja stanj na sliki 3.7-1. Uporabite pomnilne celice JK in Shefferjeva ali Pierceova vrata. Vhodna in izhodna abeceda ter stanja kodirajte po tabeli 3.7-1.



Slika 3.7-1. Diagram prehajanja stanj.

Tabela 3.7-1. Kodirne tabele.

vhodna abeceda	koda x
x_1	0
x_2	1

Izhodna Abeceda	koda z
z_1	0
z_2	1

notranja stanja	koda $y_1\ y_2$
S_0	0 0
S_1	0 1
S_2	1 0
S_3	1 1

Priprava na vajo: Doma opravite naloge iz točk 1, 2, 3, 4, 5 in 6.

1. Izpolnite vzbujevalno tabelo za pomnilno celico JK.

y	y^+	J	K
0	0		
0	1		
1	0		
1	1		

2. Izpolnite pravilnostno tabelo za diagram prehajanja stanj s slike 3.7-1:

sedanje stanje		naslednje stanje /izhod x =	
		0	1
y_1	y_2	$y_1^+ y_2^+/z$	$y_1^+ y_2^+/z$
0	0	/	/
0	1	/	/
1	0	/	/
1	1	/	/

3. Izpolnite vzbujevalno tabelo.

vhod	sedanje stanje		naslednje stanje		izhod	krmilni vhodi pomnilnih celic			
x	y_1	y_2	y_1^+	y_2^+	z	J_1	K_1	J_2	K_2
0	0	0							
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

3. Zapišite vzbujevalne enačbe krmilnih vhodov (J_1, K_1, J_2, K_2) pomnilnih celic ter izhodno enačbo (z).

J_1:

y_1,y_2 \ x	0	1
00		
01		
11		
10		

J_1 =

K_1:

y_1,y_2 \ x	0	1
00		
01		
11		
10		

K_1 =

J_2:

y_1,y_2 \ x	0	1
00		
01		
11		
10		

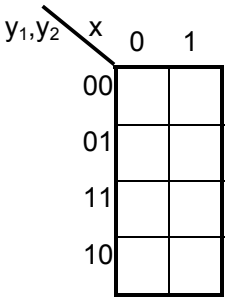
J_2 =

K_2:

y_1,y_2 \ x	0	1
00		
01		
11		
10		

K_2 =

z:



J_1 = K_1 =

J_2 = K_2 =

z =

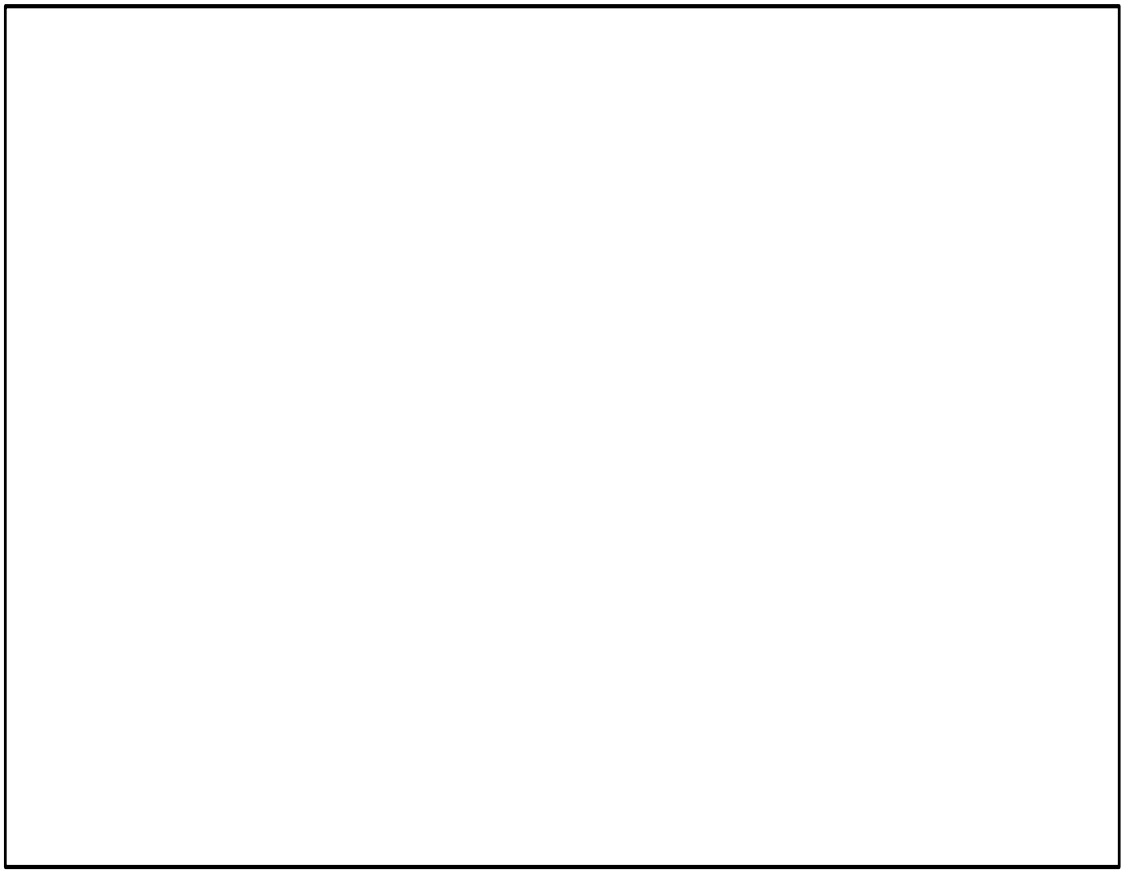
5. Vzbujevalne enačbe in izhodno enačbo izrazite s Shefferjevim funkcijsko polnim sistemom.

J_1 = K_1 =

J_2 = K_2 =

z =

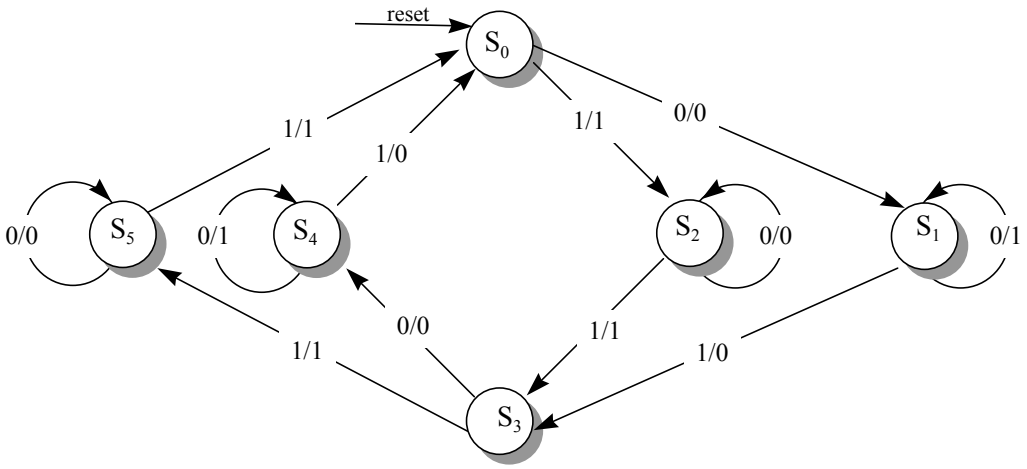
6. Narišite načrt vezja glede na enačbe v točki 5.



7. Sestavite vezje na laboratorijski plošči. Preverite, ali se delovanje sekvenčnega vezja ujema s podanim diagramom prehajanja stanj (slika 3.7-1). Pri preizkusu testirajte, ali lahko vezje zavzame vsa možna stanja. Nato preverite še, ali so možni vsi prehodi med stanji. Priključke za SET in RESET na pomnilnih celicah vežite tako, da je možno prek stikal asinhrono nastaviti notranje stanje sinhronega sekvenčnega vezja.

3.8 Minimizacija avtomata

Besedilo naloge: Na sliki 3.8-1 je z diagramom prehajanja stanj podan avtomat s šestimi stanji. Začetno stanje avtomata je $S_{0.}$. Diagram prehajanja stanj minimizirajte in narišite minimiziran diagram prehajanja stanj. Nato izvedite vse korake sinteze vezja, vezje sestavite in ga preizkusite na laboratorijski plošči.



Slika 3.8-1. Diagram prehajanja stanj.

Priprava na vajo: Doma opravite naloge iz točk 1, 2, 3, 4, 5, 6, 7 in 8.

1. Ugotovite, s koliko pomnilnimi celicami je mogoče realizirati neminimizirani avtomat s slike 3.8-1.

p =

2. Izvedite postopek minimizacije avtomata s slike 3.8-1.

3. Narišite minimiziran diagram prehajanja stanj in minimizirano tabelo prehajanja stanj.

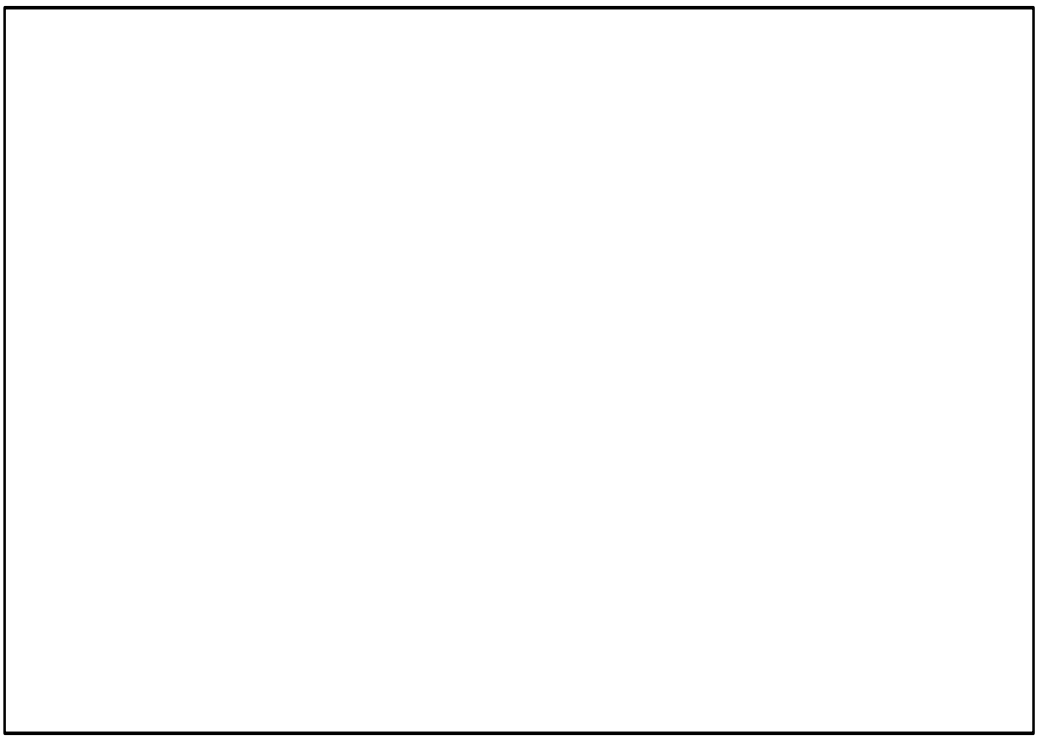
4. Kodirajte stanja.

5. Narišite kodirano tabelo prehajanja stanj.

6. Zapišite vzbujevalne enačbe in enačbo za izhod vezja.

7. Vzbujevalne enačbe in izhodno enačbo izrazite s Pierceovim funkcijsko polnim sistemom.

8. Narišite vezalno shemo avtomata glede na enačbe v točki 7.



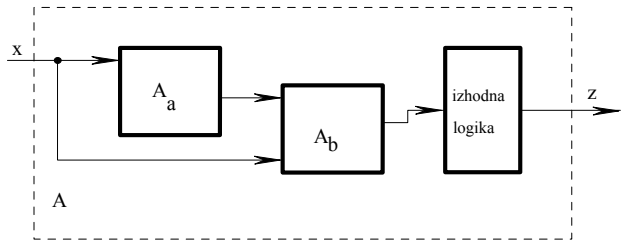
9. Sestavite vezje na laboratorijski plošči in preverite, ali se realizirani avtomat ujema s svojo specifikacijo na sliki 3.8-1. Pri realizaciji vezja vežite asinhrone vhode RESET in SET pomnilnih celic tako, da boste lahko avtomat asinhrono postavili v želeno stanje. Avtomat naj se po vklopu napajanja znajde v začetnem stanju S_0.

3.9 Dekompozicija avtomata

Besedilo naloge: Na sliki 3.9-1 je s tabelo prehajanja stanj podan Moorov avtomat A. Izvedite serijsko dekompozicijo avtomata A na podavtomata A_a in A_b po sliki 3.9-2. Za podavtomat A_a izberite particijo $\prod_a = \{\overline{1}, \overline{2,3}, \overline{4,5}\}$. Particija $\prod_b$ naj ima dva bloka.

sedanje stanje	naslednje stanje x = 0	naslednje stanje x = 1	izhod z
1	2	1	0
2	3	5	1
3	2	4	0
4	1	3	1
5	1	2	0

Slika 3.9-1. Diagram prehajanja stanj za Moorov avtomat A.



Slika 3.9-2. Blokovna shema za serijsko dekompozicijo avtomata A.

Priprava na vajo: Dobro preštudirajte postopek dekompozicije avtomatov. Doma opravite naloge iz točk 1, 2, 3, 4, 5, 6 in 7.

1. Dokažite, da ima particija $\prod_a = \{\overline{1}, \overline{2,3}, \overline{4,5}\} = \{B_1; B_2; B_3\}$ substitucijsko značilnost.

2. Izberite dvoblokovno particijo $\prod_b = \{B_4; B_5\}$ z blokoma B_4 in B_5, tako da bo izpolnjen pogoj za serijsko dekompozicijo avtomata A na podavtomata A_a in A_b: $\prod_a \bullet \prod_b = \prod_0$.

$\prod_b =$

3. Izpolnite tabeli prehajanja stanj za podavtomata A_a in A_b. V podavtomat A_b vstopa stanje $B_i, 1 \leq i \leq 3$, podavtomata A_a in vhodna črka x. Vse možne kombinacije na vhodu podavtomata A_b so: $B_1 0$, $B_1 1$, $B_2 0$, $B_2 1$, $B_3 0$ in $B_3 1$.

A_a: A_b:

sedanje stanje	naslednje stanje x = 0	naslednje stanje x = 1
B_1		
B_2		
B_3		

sedanje stanje	naslednje stanje/izhod $1 \leq i \leq 3$, $B_i x$ = $B_1 0$	$B_1 1$	$B_2 0$	$B_2 1$	$B_3 0$	$B_3 1$
B_4	/	/	/	/	/	/
B_5	/	/	/	/	/	/

4. Kodirajte stanja, podavtomatov A_a in A_b.

stanje	koda stanja y_2 y_1
B_1	0 0
B_2	0 1
B_3	1 0

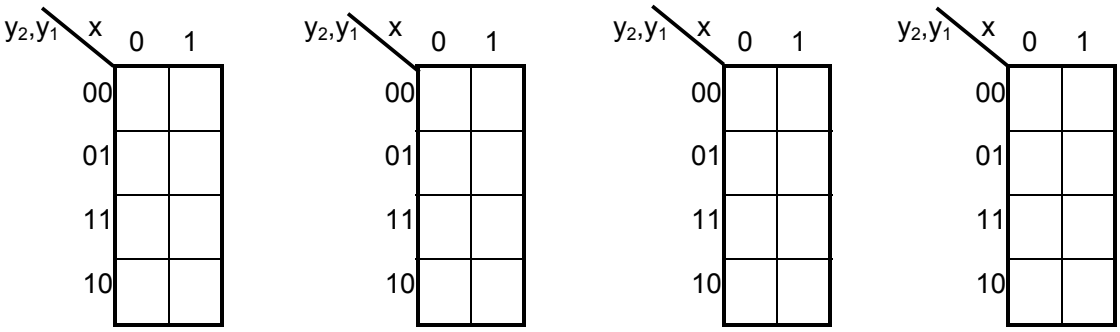
stanje	koda stanja y_0
B_4	0
B_5	1

5. Izpolnite vzbujevalno tabelo za podavtomat A_a.

vhod	sedanje stanje		naslednje stanje		krmilni vhodi pomnilnih celic			
x	y_2	y_1	y_2^+	y_1^+	J_2	K_2	J_1	K_1
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

6. Zapišite enačbe krmilnih vhodov (J_1, K_1, J_2, K_2) pomnilnih celic podavtomata A_a.

y_1^+: y_2^+:



J_1= J_2=

K_1= K_2=

7. Izpolnite vzbujevalno tabelo za podavtomat A_b.

vhod			sedanje stanje	naslednje stanje	krmilna vhoda pomnilne celice		izhod
x	y_2	y_1	y_0	y_0^+	J_0	K_0	z
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

8. Zapišite enačbe krmilnih vhodov (J_0, K_0) podavtomata A_b.

J_0:

y_1y_0 \ $x\,y_2$	0 0	0 1	1 1	1 0
00				
01				
11				
10				

K_0:

y_1y_0 \ $x\,y_2$	0 0	0 1	1 1	1 0
00				
01				
11				
10				

J_0=

K_0=

z =

9. Narišite vezalno shemo za podavtomata A_a in A_b ter izhodno logiko. Označite priključke čipov.

10. Vezje iz točke 9 sestavite na laboratorijski plošči. Najprej preverite delovanje podavtomatov A_a in A_b glede na tabeli v točkah 5 in 7. Nato preverite pravilnost delovanja celotnega avtomata A glede na njegovo specifikacijo na sliki 3.9-1.

4 NAVODILA ZA OPRAVLJANJE RAČUNALNIŠKIH VAJ

Pri računalniških vajah veljajo naslednja pravila:

- Računalniške vaje se opravijo predvidoma v treh delih.
- Prisotnost na vseh računalniških vajah je obvezna.
- Urnik vaj s seznamom za prijavo k vajam se razpiše v tednu pred pričetkom cikla računalniških vaj.
- Študent mora poznati in upoštevati vse tehnične podatke za elemente, ki jih bo pri vajah uporabljal. Tehnični podatki so zbrani v prilogi na koncu vaj.
- Uspešno opravljene računalniške vaje in pozitivno ocenjen projekt je pogoj za pristop k izpitu.

4.1 NAVODILO ZA IZDELAVO PROJEKTA

Računalniške vaje se izvajajo na osebnih računalnikih v okolju Windows. Za realizacijo nalog uporabljamo program LogicWorks 3.0.0. (http://www.capilano.com/LogicWorks/) oz. nadgradnjo. Po kratkem uvajanju v programski paket LogicWorks sledi izvedba nalog iz poglavja 3, ki jih izvajamo tudi v laboratoriju z realnimi vezji. V nadaljevanju si izberite med podanimi projekt, ki ga želite realizirati. Po zaključku dela boste projekt predstavili in zagovarjali. Po zagovoru oddajte pismeno poročilo, ki vsebuje naslednje točke:

1. Opis problema.
2. Blokovna shema vezja.
3. Opis signalov.
4. Diagrami prehajanja stanj, tabele, minimizacije in enačbe.
5. Opis uporabljenih elementov in njihovih funkcij.
6. Načrt vezja.
7. Zaključek.
8. Literatura.

Pri izdelavi poročila uporabite vsa razpoložljiva tehnična sredstva. Elemente in opise elementov, ki jih v dodatku priročnika ni, poiščite na spletnih straneh proizvajalcev:

- Motorola: http://www.design-net.com/sps/General/chips-nav.html
- Philips: http://www-us.semiconductors.philips.com/
- SGS-THOMPSON: http://www.st.com/stonline/books/index.htm.

Poročilo naj ne bo daljše od sedmih strani formata A4. Ocena vaj je poprečje ocen kakovosti in zagovora izdelanega projekta ter poročila.

5 LOGICWORKS

5.1 Uvod

Potreba po vse kompleksnejših elektronskih vezjih iz leta v leto skokovito narašča, zato je vse manjši poudarek na ročnih načrtovalskih orodjih. Načrtovalci danes uporabljajo različna računalniško podprta načrtovalska orodja. Načrtovanje strojne opreme je zato vse bolj podobno načrtovanju programske. Na tržišču se nenehno pojavljajo nova orodja za načrtovanje strojne opreme. Izbira je zelo pestra. Izbiramo lahko izmed cenejših, praviloma zelo enostavnih orodij, pa vse do profesionalnih, za uporabo bolj zapletenih in praviloma zelo dragih načrtovalskih orodij. Med uporabniki so navadno priljubljena orodja, ki so na voljo v različnih izvedbah: v izobraževalni in profesionalni. Izobraževalna se od profesionalnih navadno razlikujejo predvsem po obsegu vezij, ki jih lahko načrtujemo, cenejša programska orodja pa ponavadi tudi nimajo tako bogate knjižnice z elementi kot dražja. Običajno se po številu funkcij obe izvedbi programskih orodij ne razlikujeta. Posamezni proizvajalci načrtovalskih orodij prodajajo testne različice za simboličen znesek. Njihova uporaba pa je praviloma časovno omejena, včasih pa programi tudi ne dovolijo izvajanja nekaterih pomembnih funkcij (shranjevanje načrta, tiskanje itd.)

Pri podjetju Capilano Computing Systems so razvili programsko orodje za načrtovanje elektronskih vezij *DesignWorks*. *DesignWorks* je namenjen profesionalnemu delu. Uporabljajo ga številne državne ustanove in znana podjetja po vsem svetu. Njegova odlika je predvsem lahka uporaba ter velika hitrost izvajanja funkcij. Za izobraževalne namene so pri Capilano Computing Systems razvili še študentsko različico programa. Poimenovali so ga *LogicWorks*. Program *LogicWorks* je zelo podoben *DesignWorks*-u. Čeprav je namenjen predvsem za izobraževanje, ga lahko uporabimo tudi pri profesionalnem delu. Zaradi dostopne cene je primeren predvsem za uporabo na fakultetah, kjer zaradi omejenih finančnih sredstev nakup večjega števila dragih profesionalnih CAD programov navadno ni možen. Naštejmo nekaj dobrih lastnosti programskega paketa LogicWorks:

- primeren je za hitro uvajanje v računalniško podprto načrtovanje,
- načrtovalec lahko takoj preveri rezultate delovanja svojega vezja,
- uporaba programa je zelo preprosta,

- namestitev programa je hitra in preprosta,
- logična zasnova in zgradba programa je enostavna,
- za zahtevnejše aplikacije obstaja močnejša različica programa,
- uporabljeni so standardni simboli za elemente,
- možna je izdelava lastnih elementov in knjižnic,
- možna je izdelava seznama povezav in uporabljenih elementov v vezju.

5.2 Postopek za namestitev in zagon programa

Program namestimo na osebni računalnik, ki ima naložen Microsoftov operacijski sistem Windows95, Windows98 ali WindowsNT. Za namestitev programa potrebujemo disketi, na katerih so programski paket (verzija 3.0.0) ter knjižnice z elementi. Podjetje Capilano Computing Systems program prodaja v paketu skupaj z dobro napisanim priročnikom v angleškem jeziku. Program deluje hitro in brez omejitev tudi na malo starejših in manj zmogljivih osebnih računalnikih.

Program običajno namestimo na trdi disk C ali D v mapo LW300. Z namestitvijo dodatnih podmap (npr. DELO, PROJEKTI, VAJA itd.) pa poskrbimo za večjo preglednost pri kasnejšem shranjevanju načrtov.

Če želimo program namestiti na računalnik z operacijskim sistemom Windows98 ali WindowsNT, potrebujemo dodatno še program, ki ga lahko prenesemo z domačih strani podjetja Capilano Computing Systems (slika 5.2-1).

Slika 5.2-1. Informativno okno z osnovnimi podatki o podjetju in različici programa.

Po zagonu dodatnega programa imamo na računalniku nameščeno najnovejšo, t.j. različico 3.0.3 (16) programa LogicWorks. Za enostavnejši priklic programa je priporočljivo, da na omizju namestimo ikono z imenom LW303, ki nam omogoča bližnjico za zagon programa.

Program poženemo z dvakratnim zaporednim klikom z miško na ikono LW303 in na zaslonu se pojavi več oken. V nadaljevanju si bomo ogledali nekaj osnovnih lastnosti o oknih in razložili funkcije programa.

5.3 Okno LogicWorks

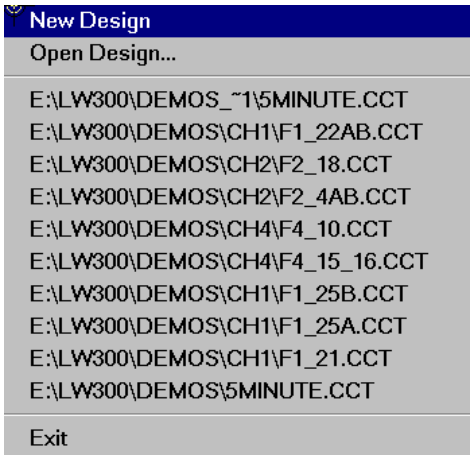
V oknu LogicWorks (slika 5.3-1) lahko izbiramo med polji **File, Tools, Window** in **Help**. S postavitvijo kazalca miške na posamezna polja odpiramo povlečne menije, kjer s postavitvijo kazalca miške izberemo želen ukaz.

Slika 5.3-1. Okno LogicWorks.

5.3.1 Polje **File**

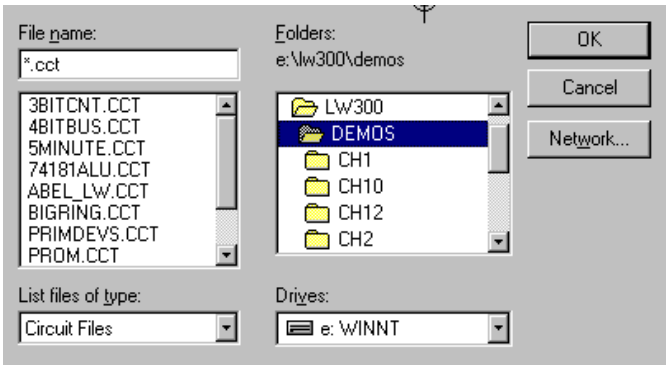
V polju **File** (slika 5.3-2) lahko na začetku izbiramo med naslednjimi ukazi:

- *New Design,*
- *Open Design,*
- *Exit.*



Slika 5.3-2. Povlečni meni polja **File**.

Če izberemo ukaz ***New Design***, se nam na zaslonu pojavijo tri okna: okno **Drawing** za risanje novega vezja (privzame ime design1.cct), okno za izbiranje elementov iz knjižnic **Parts** ter okno za simulacijo **Timing**. Če izberemo ukaz ***Open Design***, se odpre okno (slika 5.3-3), v katerem izberemo enega izmed že shranjenih načrtov vezja (npr. PROM.CCT).



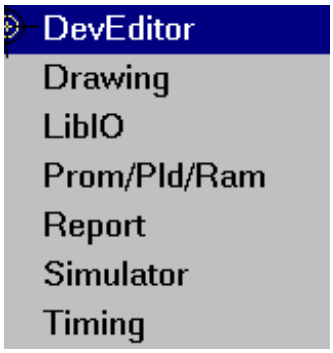
Slika 5.3-3. Okno za izbiro med shranjenimi načrti.

V delu menija **File** je zapisanih tudi do deset imen nazadnje shranjenih vezij ter poti do njih. Izbrano vezje prikličemo v okno **Drawing**.

Z operacijo **Exit** zapustimo program.

5.3.2 Polje **Tools**

V polju **Tools** (slika 5.3-4) lahko izbiramo med ukazi, ki aktivirajo posamezne programske sklope programa LogicWorks. Z ukazi dejansko odpiramo okna na zaslonu, ki pripadajo naštetim programom.

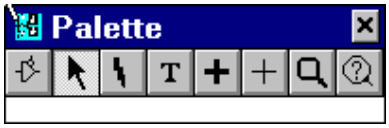


Slika 5.3-4. Ukazi v polju **Tools**.

Program LogicWorks omogoča, da je na zaslonu odprtih istočasno več oken. Za učinkovito delo v nekaterih oknih morajo le-ta imeti neko minimalno velikost. Ta je seveda odvisna tudi od velikosti ekrana in od izbrane nastavitve njegove ločljivosti, zato okna programov, ki jih nujno ne potrebujemo, začasno raje zapremo. Z ukazi polja Tools aktiviramo naslednja okna:

- *DevEditor,*
- *Drawing,*
- *LibIO,*
- *Prom/Pld/Ram,*
- *Report,*
- *Simulator,*
- *Timing.*

DevEditor je programsko okno za oblikovanje in vključevanje novih ter spreminjanje obstoječih logičnih elementov v knjižnicah. **Drawing** je programsko okno za risanje vezij (na začetku program privzame ime design1.cct). Hkrati se s programskim oknom **Drawing** aktivira tudi okno **Parts** za izbiranje elementov iz knjižnic ter programsko okno **Timing** za prikaz časovnih signalov. Z aktiviranjem okna **Drawing** se samodejno aktivira še okno **Pallete** (slika 5.3-5). Okno **Pallete** vsebuje orodja, ki jih potrebujemo za risanje vezij v programskem oknu **Drawing**.

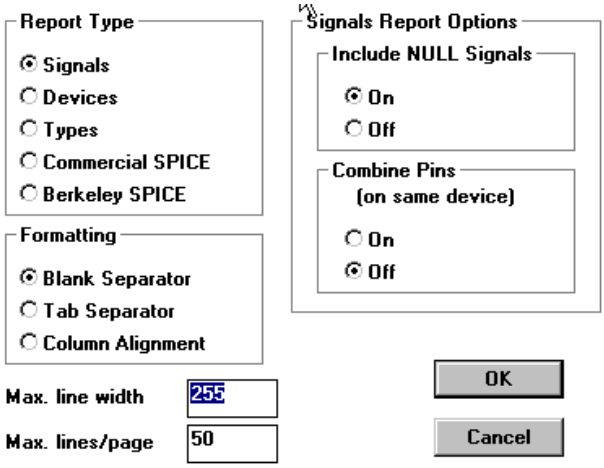


Slika 5.3-5. Okno za izbiro funkcije kurzorja.

Z izbiro ukaza **LibIO** aktiviramo okno **Parts** s knjižnicami elementov. Samodejno se naložijo vse tiste knjižnice, ki se nahajajo v inicializacijski datoteki LW.ini.

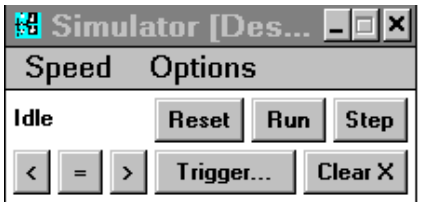
Z izbiro ukaza **Prom/Pld/Ram** aktiviramo programsko okno, ki je namenjeno generiranju programirljivih logičnih vezij, kot so RAM, PLA in ROM.

Z izbiro ukaza **Report** aktiviramo programsko okno (slika 5.3-6), ki nam da možnost izpisov seznama uporabljenih elementov, seznama povezav ter modelov za analogni simulator SPICE.



Slika 5.3-6. Okno za izbiro izpisa različnih informacij o vezju.

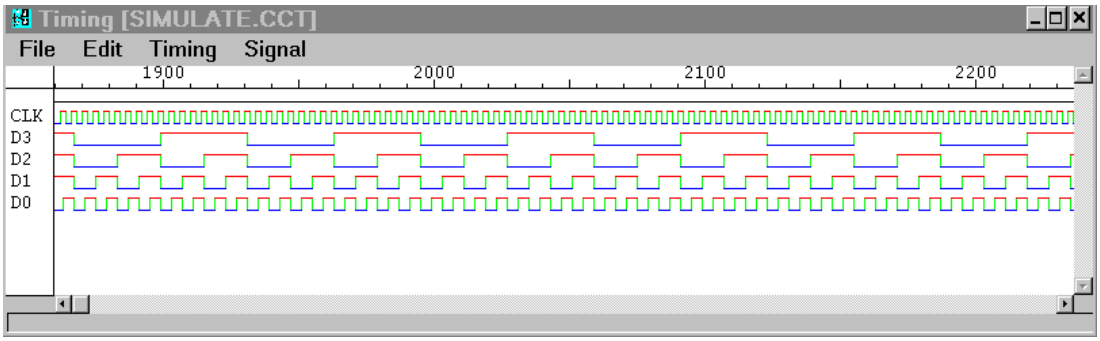
Z izbiro ukaza **Simulator** aktiviramo okno (slika 5.3-7), v katerem nastavljamo različne parametre za simulacijo, katere potek vidimo v časovnem oknu **Timing** (slika 5.3-8).



Slika 5.3-7. Okno, v katerem spreminjamo parametre simulatorja.

V oknu **Simulator** z izbiro polja **Speed** odpremo okno, kjer lahko nastavimo hitrost simulacije (Stop, Crawl, Ooze, ..., Jog, Run), z izbiro polja **Options** nastavljamo časovne parametre elementov in pripnemo določeni logični nivo k izbrani povezavi v oknu **Drawing**. Z gumbom **Reset** zbrišemo dosedanji izris poteka simulacije, z gumbom **Run** pa jo poženemo z maksimalno možno hitrostjo. Gumb **Step** omogoča proženje simulacije po posameznih korakih. Z gumbi **< = >** določimo časovno širino okna, s katerim bomo opazovali potek simulacije. S pritiskom na gumb **<** časovno periodo po korakih zmanjšujemo, s pritiskom na gumb **>** pa jo korakoma zvišujemo. S pritiskom na gumb **=** se časovno okno nastavi na neko privzeto vrednost opazovane periode. Z gumbom **Trigger** nastavljamo proženje na določen dogodek. Določimo lahko, da nas zvočni signal npr. opozori na začetek ali konec nekega dogodka.

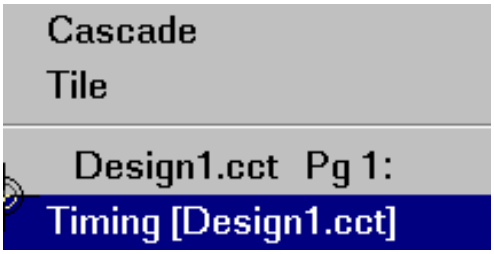
Dogodke oz. spremembe logičnih nivojev signalov v vezju opazujemo (slika 5.3-8) v programskem oknu **Timing** (pa tudi na nekaterih elementih v vezju) in so vidni le, če je simulator aktiven (simulacija teče). Logični nivoji v vezju so prikazani z različnimi barvami in vzorci. Zaporedje izrisa signalov je določeno z njihovim zaporedjem vpisa v oknu **Drawing**, lahko pa ga enostavno spremenimo tako, da s kazalcem miške signal označimo in ga potegnemo na želeno drugo mesto.



Slika 5.3-8. Časovno okno, v katerem opazujemo časovne poteke signalov v vezju.

5.3.3 Polje **Window**

V polju **Window** (slika 5.3-9) lahko izberemo med dvema načinoma razpostavitve oken na zaslonu: kaskadnim in vzporednim. Pri izbiri kaskadnega načina (**Cascade**) se posamezna okna namestijo drugo preko drugega, tako da je zadnje aktivno okno postavljeno na vrh kaskade, za preostala pa je vidna le ukazna vrstica. Preostala okna enostavno prikličemo na vrh tako, da jih aktiviramo z enojnim klikom miške na ukazno vrstico. Pri vzporednem načinu (**Tile**) se odprta okna po ekranu razvrstijo drugo ob drugem. Slabost slednjega načina je, da se z večanjem števila na zaslonu odprtih oken velikost posameznih zmanjšuje. V spodnjem delu polja **Window** so navedena vsa odprta okna in do želenega lahko pridemo tudi neposredno.



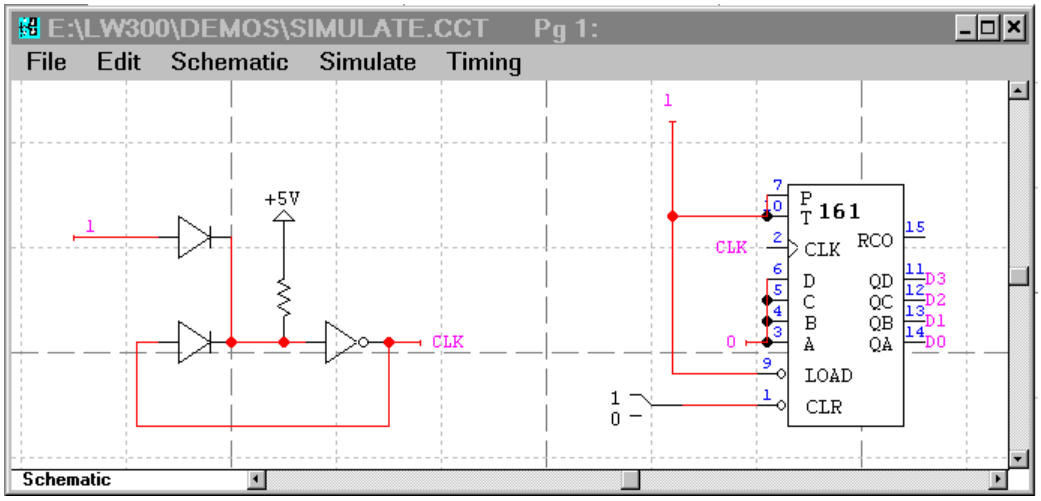
Slika 5.3-9. Okno, v katerem izberemo razvrstitev oken na zaslonu.

5.3.4 Polje **Help**

Z izbiro polja **Help** dobimo podatke o proizvajalcu. Napotkov za uporabo programskega paketa polje ne podpira.

5.4 Okno Drawing

V oknu Drawing (slika 5.4-1) izbiramo med polji **File, Edit, Shematic, Simulate** in **Timing**. V tem oknu snujemo logično vezje ali podvezje.



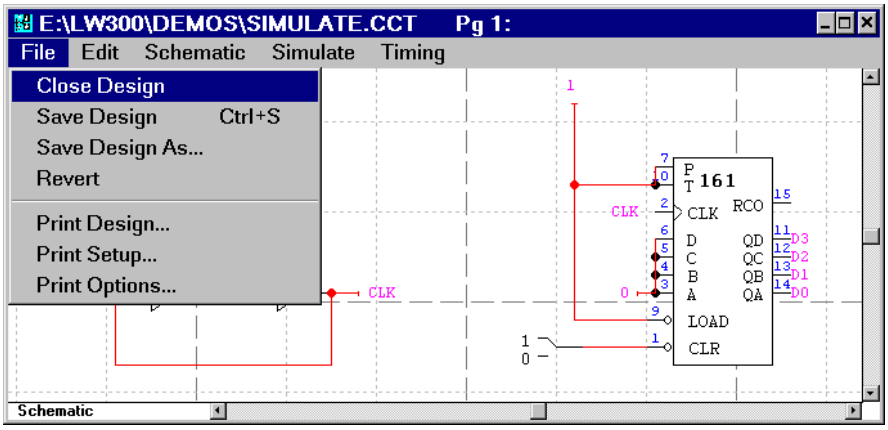
Slika 5.4-1. Okno Drawing, v katerem snujemo digitalna vezja.

5.4.1 Polje **File**

Z izbiro polja **File** (slika 5.4-2) odpremo meni, v katerem izberemo želeno operacijo za odprto delovno okolje v oknu Drawing.

Polja **Close Design, Save Design, Save Design As...** in **Revert**

Odprto delovno okno zapremo z izbiro ukaza **Close Design**, vezje shranimo z ukazom **Save Design**. Če želimo zapreti okno in so po zadnjem shranjevanju nastale spremembe v vezju, nas bo program pri izhodu na to opozoril. Če želimo vezje shraniti pod novim imenom, uporabimo ukaz **Save Design As...** . Če želimo izvorno datoteko ponovno včitati, izberemo polje **Revert**.



Slika 5.4-2. Aktiviran meni **File**.

Polja **Print Design..., Print Setup...** in **Print Options...**

Vsebino odprtega delovnega okna tiskamo z ukazom **Print Design...**. Z ukazom **Print Setup...** se odpre okno, v katerem nastavimo želene parametre tiskalnika in s **Print Options...** nastavimo korekcijski faktor povečave/pomanjšave delovnega okolja glede na tiskano.

5.4.2 Polje **Edit**

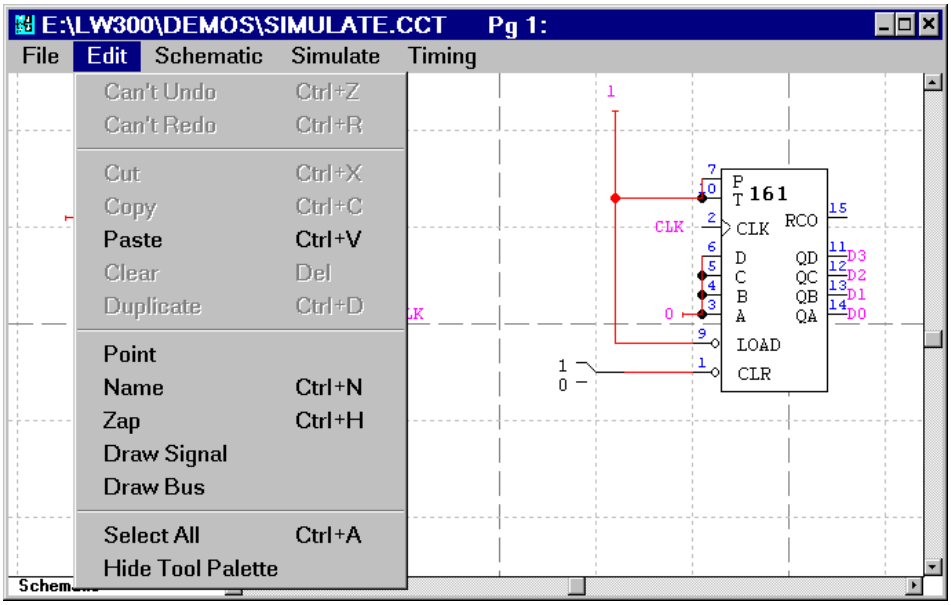
Z izbiro polja **Edit** (slika 5.4-3) odpremo meni, v katerem izberemo enega izmed ukazov **Undo, Redo Cut, Copy, Paste, Clear, Duplicate, Point, Name, Zap, Draw Signal, Draw Bus**, **Select All** ter **Hide Tool Palette**

Polja **Undo, Redo**

Z ukazom **Undo** prekličemo nazadnje izveden ukaz v oknu Drawing. Z ukazom **Redo** prekličemo ukaz **Undo**.

Polja **Cut, Copy, Paste, Clear** in **Duplicate**

Z ukazom **Cut** brišemo posamezne elemente, povezave ali skupino elementov in povezav v delovnem prostoru. Z ukazom **Copy** kopiramo posamezne elemente, povezave ali skupino elementov in povezav v delovnem prostoru. Z ukazom **Paste** vstavljamo posamezne elemente, povezave ali skupino elementov in povezav v delovnem prostoru. Z ukazom **Duplicate** podvajamo posamezne elemente, povezave ali skupino elementov in povezav v delovnem prostoru.



Slika 5.4-3. Ukazi polja Edit.

Polja **Point, Name, Zap, Draw Signal** in **Draw Bus**

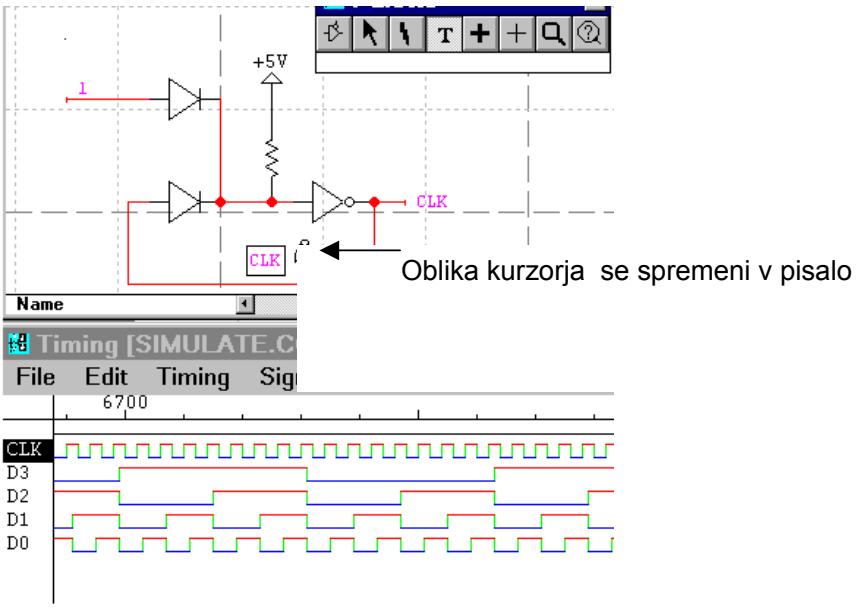
Z ukazi **Point, Name, Zap, Draw Signal** in **Draw Bus** spreminjamo namembnost kazalca. Ukazi so enakovredni ukazom, ki jih lahko prikličemo v oknu z orodji **Palette.**

Z aktiviranjem ukaza **Point** ( ) je kazalec v obliki puščice, kar predstavlja običajen način delovanja, njegova funkcija pa je označevanje elementov in povezav v vezju ter njihovo povezovanje. Ukaz lahko aktiviramo tudi s potrditvijo gumba v obliki puščice v oknu **Palette** . Če želimo povezati dva priključka v vezju, postavimo vrh puščice na mesto začetka povezovanja, pritisnemo levo tipko miške, jo držimo do mesta, kjer se

povezava konča, in nato spustimo tipko miške. Povezavo lahko tvorimo tudi v več zaporednih korakih.

Ukaz **Name** () aktiviramo, kadar želimo na poljubno mesto na delovni površini vnesti neko besedilo. Po potrditvi ukaza kazalec dobi obliko pisala (slika 5.4-4). Z miško postavimo pisalo na želeno mesto delovne površine in pritisnemo levo tipko. Pojavi se okvir, v katerega nato vtipkamo besedilo. Operacijo končamo s pritiskom na tipko ENTER. Z ukazom **Name** pa označujemo tudi povezave v vezju. Z miško pisalo postavimo na izbrano povezavo in pritisnemo levo tipko. Pojavi se okvir, v katerem je neko že privzeto ime povezave; tega nato prepišemo z lastnim imenom in končamo s pritiskom na tipko ENTER. Besedilo, ki ponazarja ime povezave, dobi vijoličasto barvo (navadno besedilo je črne barve), istočasno pa se ime povezave pojavi tudi v vrstici časovnega okna (slika 5.4-4).

Z izbiro ukaza **Zap** ali gumba  v orodni vrstici **Palette** dobi kazalec funkcijo brisanja; spremeni pa se v poševno zlomljeno črto. Velja pravilo, da element, posamezne dele povezav, imena povezav ali besedilo brišemo z enkratnim pritiskom na levo tipko miške. Če pomotoma zbrišemo kak element v vezju, lahko brisanje prekličemo z ukazom **Undo**.



Slika 5.4-4. Označevanje imena signalne povezave.

Signale in vodila praviloma tvorimo z uporabo ukaza **Draw Signal** oz. **Draw Bus**. Ukaza aktiviramo tudi z izbiro gumbov  oz.  v orodnem oknu **Palette**. Postopek za risanje

povezav je z uporabo orodij **Draw Signal** in **Draw Bus** skoraj enak, vendar je povezave (vodila) z večjim številom lomljenih črt enostavneje risati z **Draw Bus**. Če rišemo signal z orodjem **Draw Signal**, z enojnim klikom miške postavimo vogal, z dvojnim pa povezavo zaključimo.

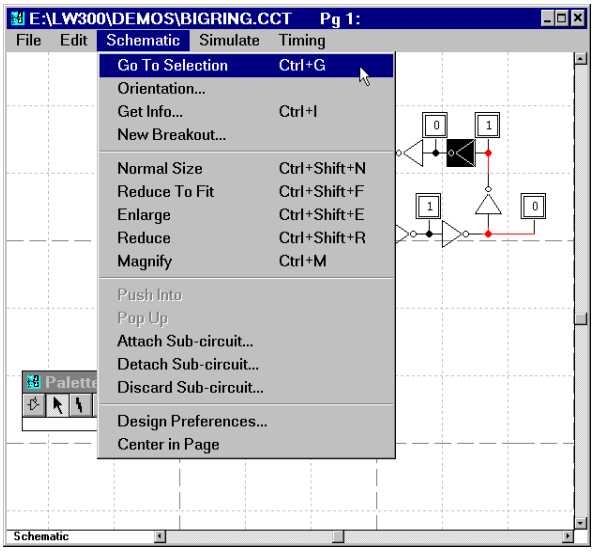
V orodni vrstici Palette imamo še tri gumbe, ki imajo naslednje funkcije:

- z gumbom  določamo smer postavitve oz. orientacijo elementa, ki ga želimo postaviti na delovni prostor.
- z gumbom  povečujemo/zmanjšujemo pogled (spreminjamo velikost izseka delovne površine).
- z gumbom  spremenimo kazalec miške v sondo, s katero lahko opazujemo vrednosti signalov kar v delovnem prostoru. Sondo postavimo na določeno povezavo in pritisnemo levo tipko miške. V sondi se pojavi trenutna vrednost signala (1, 0, X, Z ali C).

Z izbiro polja **Select All** izberemo vse elemente v delovnem polju. Z izbiro polja **Hide Tool Palette** skrijemo okno z orodji.

5.4.3 Polje **Shematic**

Z izbiro polja **Shematic** odpremo meni, v katerem izberemo želen ukaz (slika 5.4-5).



Slika 5.4-5. Meni polja Shematic.

Polje **Go To Selection**

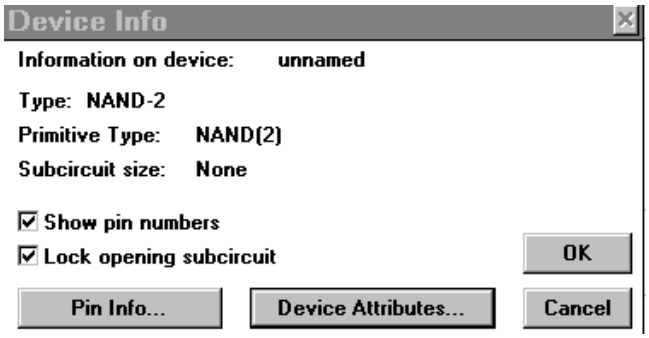
Z izbiro ukaza **Go To Selection** se v sredino zaslona preslika element ali skupina elementov, ki smo jih označili.

Polje **Orientation...**

Z ukazom **Orientation...** podobno kot z gumbom  določamo smer postavitve oz. orientacijo elementa, ki ga želimo postaviti na delovni prostor.

Polje **Get Info...**

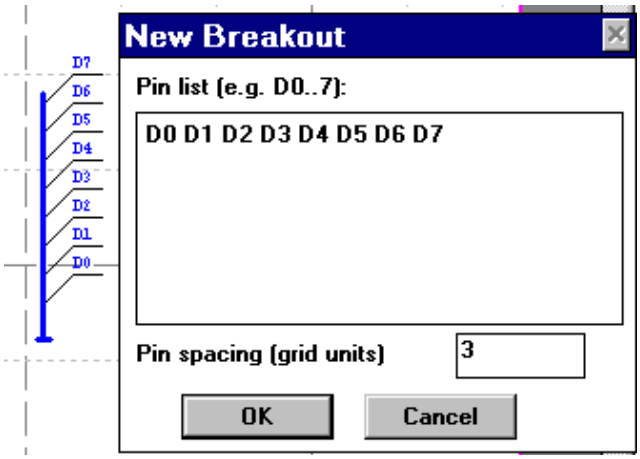
Z izbiro ukaza **Get Info...** prikličemo na zaslon informacijo o izbranem elementu (slika 5.4-6). Vklopimo/izklopimo lahko prikaz vhodno/izhodnih priključkov elementa in odklenemo/zaklenemo možnost prikaza notranje zgradbe elementa (v kolikor ta ni osnovni element).



Slika 5.4-6. Prikaz informacij o elementu in izbire nastavitve njegovih osnovnih parametrov.

Polje **New Breakout...**

Z ukazom **New Breakout...** (slika 5.4-7) določimo vstopne oz. izstopne priključke vodila.



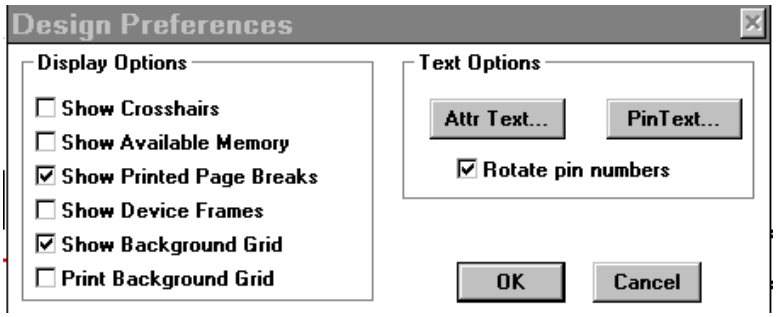
Slika 5.4-7. Odcepi D0..D7 na vodilu.

Polja **Normal Size, Reduce To Fit, Enlarge, Reduce, Magnify**

Ukaz **Normal Size** postavi velikost okna na normalno (optimalno) ločljivost, ukaz **Reduce To Fit** pa sliko polja pomanjša na takšno vrednost, da so v oknu vsi elementi nekega vezja v oknu Drawing. Ukaz **Enlarge** povečuje velikost pogleda in **Reduce** ga zmanjšuje. Ukaz **Magnify** prikliče na zaslon orodje, s katerim lahko velikost okna dinamično spreminjamo.

Polje **Design Preferences...**

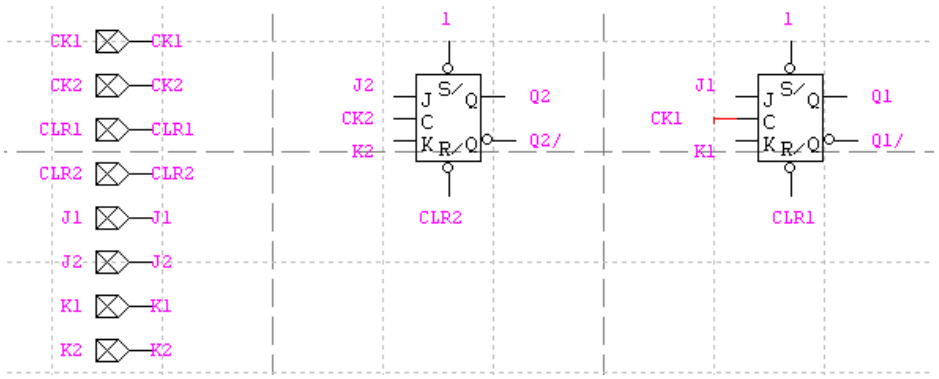
Ukaz nam na zaslon prikliče okno (slika 5.4-8), v katerem je mogoče spreminjati nekatere parametre vezja, kot so oblika kazalca, prikaz razpoložljivosti pomnilnega prostora, območje strani tiskanja, prikaz mreže.



Slika 5.4-8. Možnost spreminjanja posameznih parametrov v vezju.

5.5 Okno DevEditor

DevEditor je samostojno programsko orodje, s katerim popravljamo, prirejamo ali snujemo nove logične elemente.

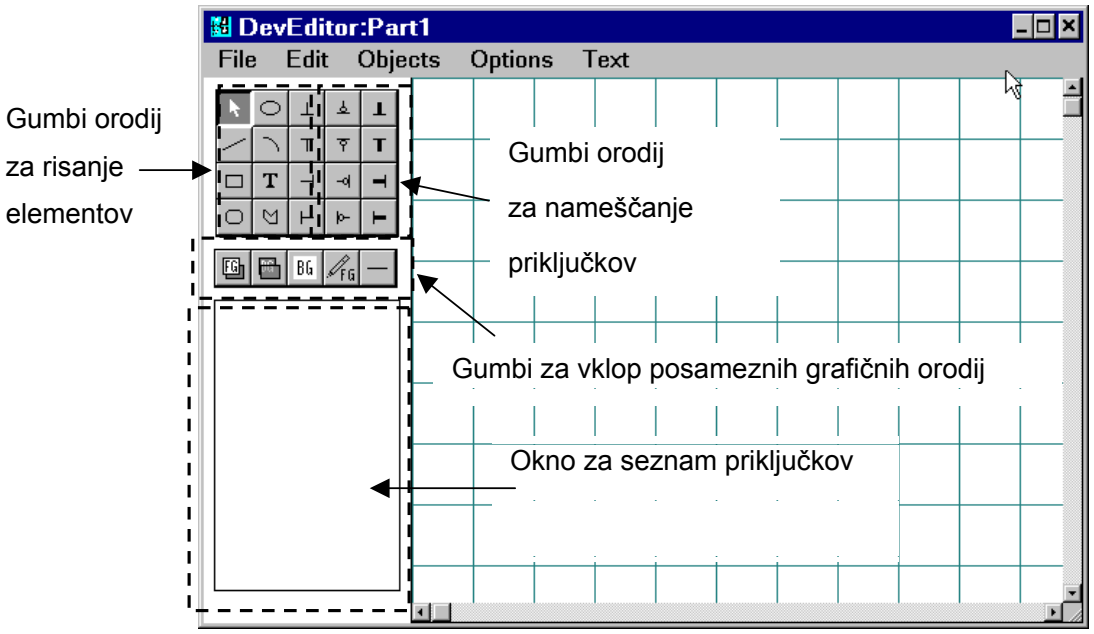


Slika 5.5-1. Konstrukcija novega logičnega elementa se prične v oknu Drawing.

Nov element v knjižnico vključimo tako, da najprej v oknu Drawing (slika 5.5-1) narišemo njegovo logično strukturo, ustrezno poimenujemo zunanje priključke in ga vključimo z orodjem DevEditor v ustrezno knjižnico.

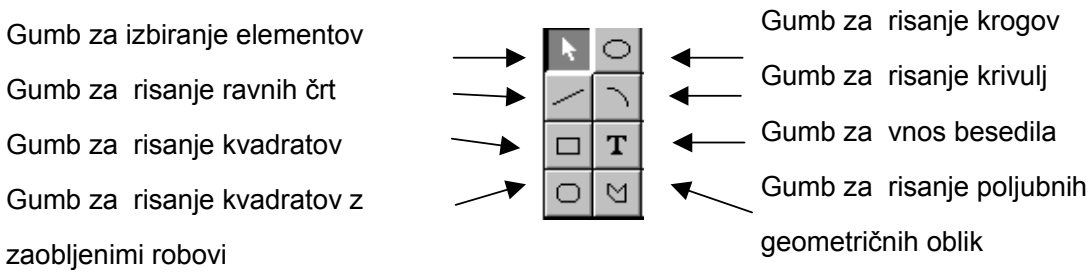
Programsko okno DevEditor (slika 5.5-2) je razdeljeno na naslednja področja:

- naslovno okno z imenom elementa, katerega trenutno načrtujemo,
- ukazno okno,
- okno z orodji,
- risalna deska,
- okno s seznamom in imeni priključkov.



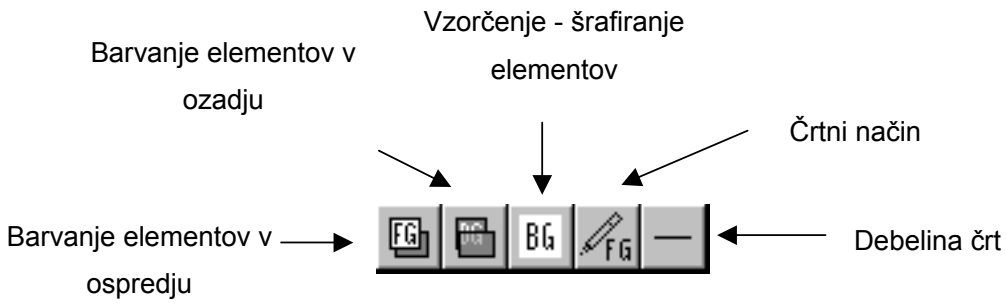
Slika 5.5-2. Okno DevEditor.

5.5.1 Orodja za risanje elementov



Slika 5.5-3. Gumbi posameznih orodij v oknu DevEditor.

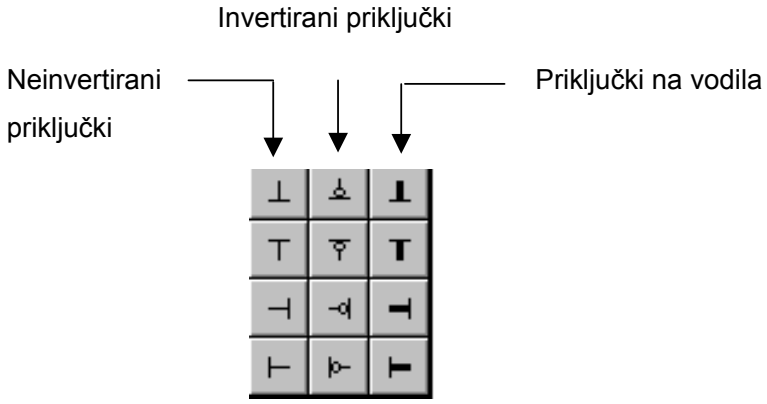
5.5.2 Grafična orodja



Slika 5.5-4. Grafična orodja v oknu DevEditor.

Izbrane elemente lahko obarvamo s poljubno barvo; pri tem ločimo barvanje elementov v ospredju in ozadju (slika 5.5-4). Orodje za vzorčenje uporabimo, če izbrane kvadrate, kroge ali kakšne druge oblike elementov želimo napolniti z nekim vzorcem. Če želimo obrobiti element z vzorčasto črto, izberemo orodje za nastavitev vzorca črt. Z zadnjim gumbom lahko aktiviramo orodje za nastavitev debeline črt.

5.5.3 Orodja za priključke elementov



Slika 5.5-5. Priključki na element.

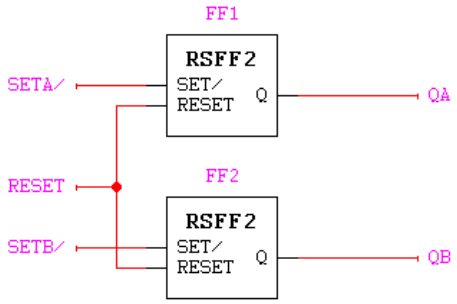
Priključki v sredinski koloni (slika 5.5-5) gumbov imajo samo grafični pomen, ne pa tudi logičnega. Če želimo, da je signal dejansko invertiran, mu dodamo še ustrezni element za negacijo signala.

5.6 Podvezja

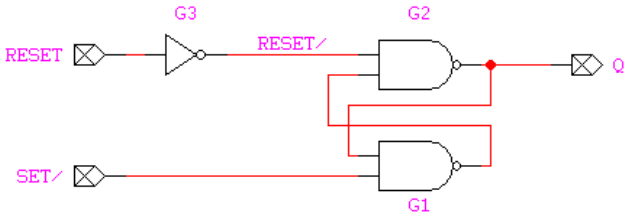
Programski paket LogicWorks nam omogoča, da načrtujemo vezja od spodaj navzgor. Vezja načrtamo najprej na nižjem nivoju, le-ta pa so podvezja vezju na višjem nivoju. Obstoječe vezje višjega nivoja pa je lahko ponovno podvezje nekega še bolj kompleksnega vezja. S tem načinom pridobimo predvsem na preglednosti pri načrtovanju, pogosto ponavljajoče se elemente pa enostavno podvajamo (primer: večbitni popolni seštevalnik). Podvezja imenujemo tudi vgnezdena vezja.

5.6.1 Primer izdelave podvezja

Poglejmo primer, kako izdelati preprosto podvezje na dveh nivojih. Na sliki 5.6-1 imamo vezje, ki ima na glavnem (višjem, prvem) nivoju dve pomnilni celici RS, na sliki 5.6-2 pa je podvezje (nižji, drugi nivo), v katerem je predstavljena dejanska logična vsebina posamezne pomnilne celice RS. V podvezju smo uporabili tri osnovne - primarne elemente (primitive). Priključki podvezja ("port connector") so logična povezava med vhodi/izhodi podvezja s priključki ("pin") osnovnega elementa.

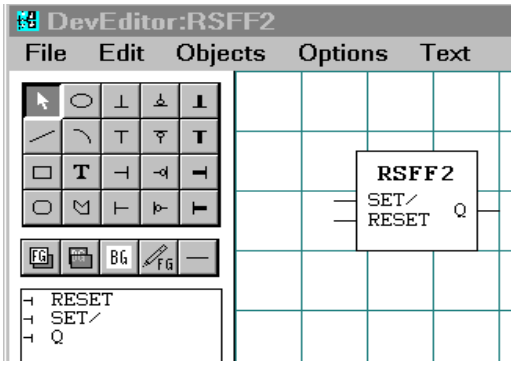


Slika 5.6-1. Vezje z dvema pomnilnima celicama RS.



Slika 5.6-2. Notranja logična zasnova pomnilne celice RS.

5.6.2 Postopek



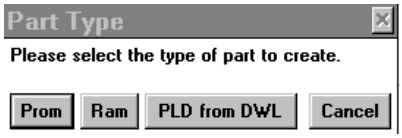
Slika 5.6-3. Izgled programskega okna DevEditor pri načrtovanju elementa RSFF2**.**

Najprej snujemo logično zasnovo podvezja v oknu **Drawing** in preverimo njegovo logično obnašanje. Za zunanje priključke podvezja uporabimo vhodno/izhodne elemente ("Port In" oz. "Port Out"), ki jih najdemo v knjižnici CONNECT.CLF. Podvezje shranimo s poljubnim imenom, npr. RSFF.CCT. Odpremo programsko okno za načrtovanje elementov **DevEditor** (slika 5.6-3). Načinov, kako oblikujemo element, je več. Oglejmo si enega. Izberemo orodje za risanje kvadratov in v oknu narišemo primerno velik kvadrat, ki predstavlja ohišje elementa. Nato izberemo gumb za postavitev priključkov na ohišje in le-tem dodamo imena (npr. RESET). V oknu, kjer so našteti priključki, se nov priključek samodejno pripiše. Določiti moramo tudi, ali gre za vhodne, izhodne ali dvosmerne priključke. Imena priključkov so obvezno ista kot imena vhodno/izhodnih priključkov v podvezju. Elementu dodamo še ime (npr. RSFF2) in ga nato shranimo v eno izmed izbranih knjižnic.

5.7 Programirljiva vezja

S programskim paketom LogicWorks je mogoče načrtovati tudi programirljiva vezja, kot so **RAM, PROM** in **PLD**. Vezja PROM in PLA tvorimo kot preproste pomnilnike oz. s polji vrat IN in ALI. Osnovni modeli elementov ne vsebujejo elementov, kot so registri, povratne zanke ali tristanjski zadrževalniki.

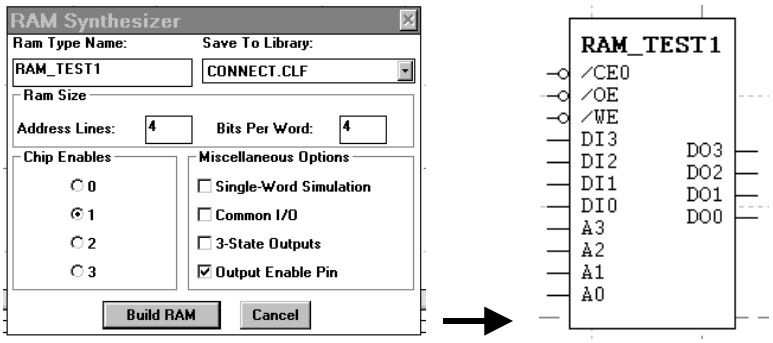
5.7.1 Tvorjenje RAM-a



Slika 5.7-1. Okno Part Type.

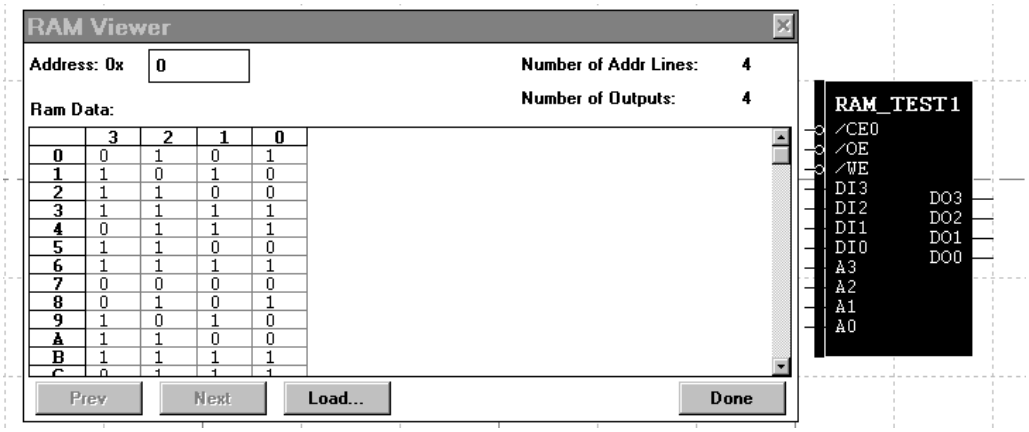
Z ukazom **Tools/Prom/PLD/RAM** na zaslon prikličemo okno **Part Type** (slika 5.7-1), kjer z gumbom **Ram** aktiviramo okno **RAM Synthesizer** (slika 5.7-2). V oknu **RAM Synthesizer** moramo določiti naslednje parametre: število naslovnih linij, širino besede in knjižnico, v katero naj se element shrani.

Na sliki 5.7-2 je primer, ki kaže, kako izdelamo RAM, ki ima 4 naslovne linije, 4-bitno širino besede ter eno linijo za izbiro elementa (CE). Element bomo shranili v knjižnico CONNECT.CLF. Ko vpišemo vse podatke, pritisnemo gumb **BuildRAM** in vezje RAM se tvori samodejno.



Slika 5.7-2. Vsebini okna RAM Synthesizer ustreza vezje RAM_TEST1.

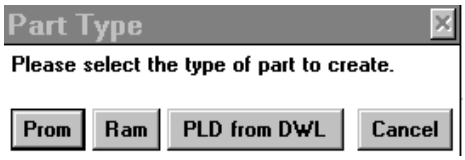
Če iz knjižnice pokličemo element RAM in ga označimo z ukazom **Tools/Prom/PLD/RAM,** na zaslon prikličemo okno **RAM Viewer** (slika 5.7-3), ki nam prikazuje binarne vrednosti besede za posamezne naslove. Vsebino RAM-a lahko določimo z datoteko formata *.hex, ki jo včitamo z izbiro gumba **Load...**.



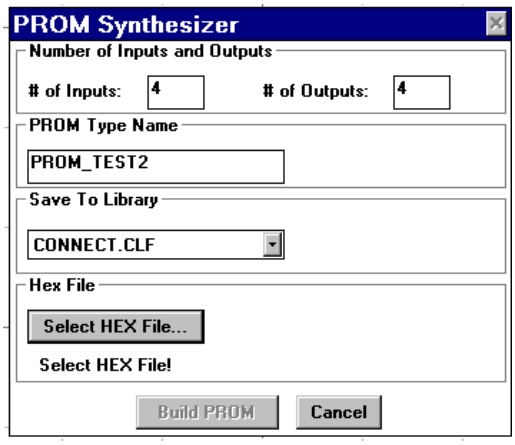
Slika 5.7-3. RAM_TEST1 in njegova vsebina.

5.7.2 Tvorjenje PROM-a

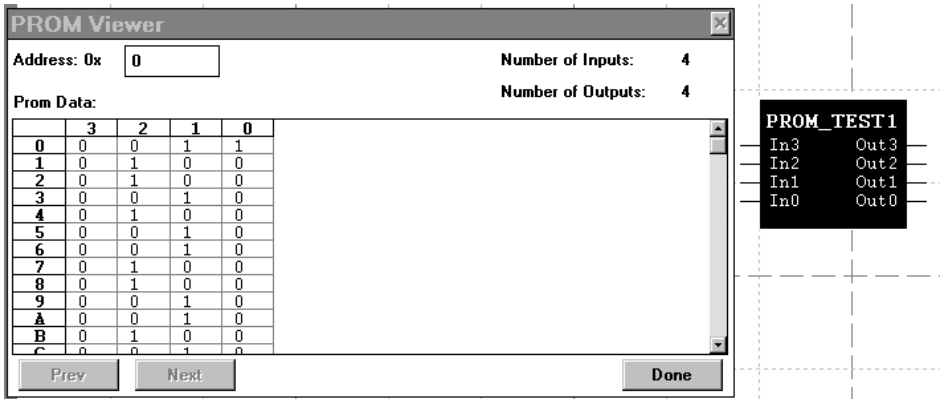
Z ukazom **Tools/Prom/PLD/RAM** na zaslon prikličemo okno **Part Type** (slika 5.7-4), kjer z gumbom **Prom** aktiviramo okno **PROM Synthesizer** (slika 5.7-5). V oknu **Prom Synthesizer** moramo določiti naslednje parametre: število naslovnih linij, širino besede, knjižnico, v katero naj se element shrani, ter datoteko HEX, ki določa vsebino PROM-a. Element bomo shranili v knjižnico CONNECT.CLF. Ko vpišemo vse podatke, pritisnemo gumb **BuildPROM** in vezje PROM se tvori avtomatično. Na sliki 5.7-6 je primer, ki kaže vsebino PROM-a s 4 naslovnimi linijami, 4-bitno širino besede ter eno povezavo za izbiro elementa (CE).



Slika 5.7-4. Okno Part Type.

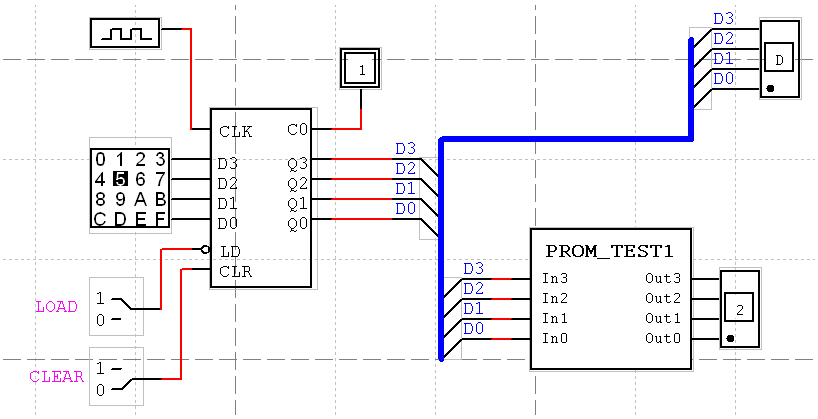


Slika 5.7-5. Okno PROM Synthesizer.



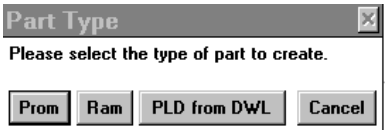
Slika 5.7-6. PROM_TEST1 in njegova vsebina.

Na sliki 5.7-7 vidimo vgradnjo PROM-a v vezje.



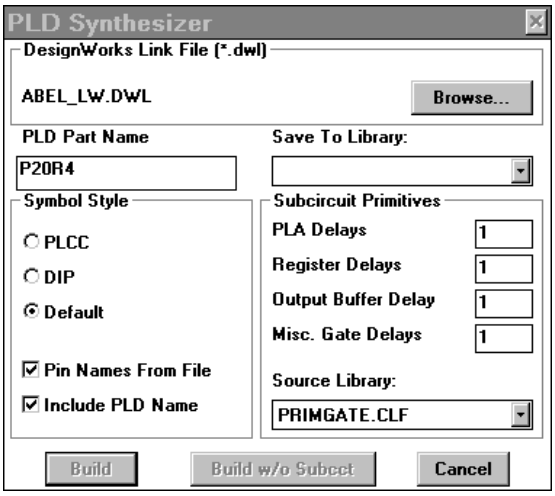
Slika 5.7-7. Primer uporabe vezja PROM_TEST1.

5.7.3 Tvorjenje PLD-jev



Slika 5.7-8. **Okno Part Type.**

Elemente PLA/PLD tvorimo podobno, kot smo prej elemente RAM in PROM. Z ukazom **Tools/Prom/PLD/RAM** na zaslon prikličemo okno **Part Type** (slika 5.7-8)**,** kjer z gumbom **PLD from DWL** aktiviramo okno **PLD Synthesizer** (slika 5.7-9). V oknu **PLD Synthesizer** izberemo datoteko ***.dwl.** Z datoteko opisujemo logično strukturo elementa PLD ***.dwl**, generiramo pa jo s programskim paketom ABEL.



Slika 5.7-9. Nastavitve parametrov za elemente PLA/PLD.

6 PROJEKTI ZA RAČUNALNIŠKE VAJE

6.1 Projekt 1: Avtomatizacija parkirišča

Besedilo naloge

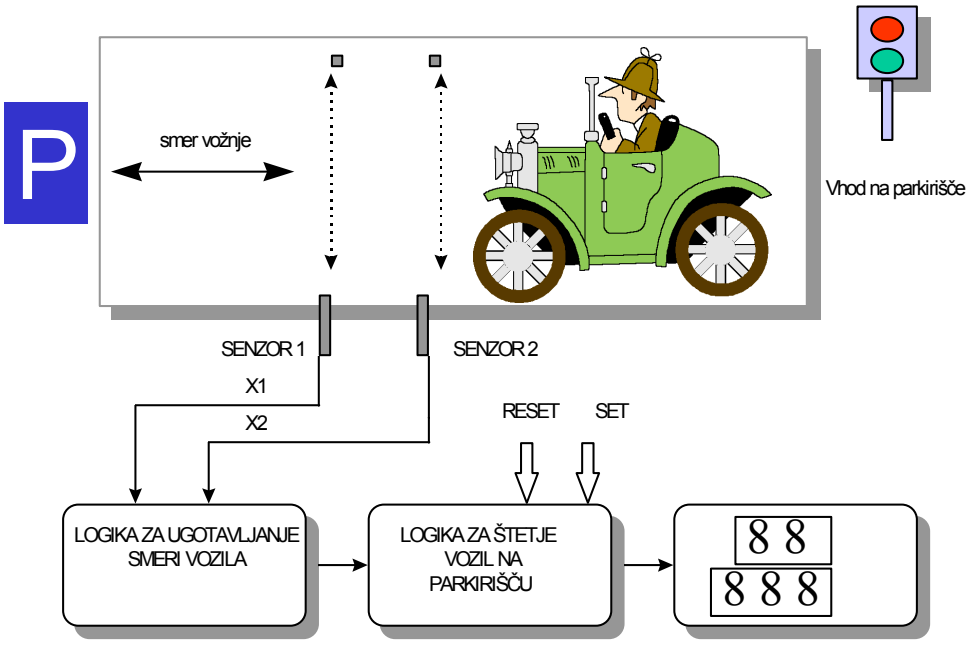
Nalogo sestavljajo naslednji deli:

- sinteza vezja za ugotavljanje smeri prečkanja kontrolnega mesta,
- sinteza števnikov, ki prikazujeta trenutno in celotno število parkiranih vozil v izbranem obdobju,
- sinteza vezja za krmiljenje semaforja (parkirišče je zasedeno/prosto),
- sinteza vezja za nastavljanje (set) in brisanje (reset) dnevnega števnika.

Prvi del naloge realizirajte s sinhronim sekvenčnim vezjem, ki ugotavlja smer, v kateri bo avtomobil prevozil kontrolno mesto parkirišča. Na kontrolnem mestu sta nameščena dva senzorja (fotocelici), ki sta med seboj oddaljena za razdaljo, ki je manjša od dolžine avtomobilov (slika 6.1-1). Signale senzorjev označimo z x_1 in x_2. Ob prekinitvi svetlobnega snopa senzorja se izhodni signal senzorja (x_1 ali x_2) postavi na 1, če pa je svetlobni snop neprekinjen, je izhodni signal senzorja (x_1 ali x_2) na 0. Predpostavimo, da kontrolno mesto lahko istočasno prevozi le eno vozilo iz poljubne smeri. Vezje ima dva izhoda (tabela 6.1-1). Izhod vezja Z_2 se postavi na 1, ko vozilo na parkirišče pripelje, in Z_1 se postavi na 1, ko vozilo odpelje s parkirišča.

Tabela 6.1-1. Tabela izhodnih signalov.

Smer vozila	Izhodna signala vezja za identifikacijo smeri Z_1 Z_2
vozilo je na parkirišče pripeljalo	0 1
vozilo je s parkirišča odpeljalo	1 0



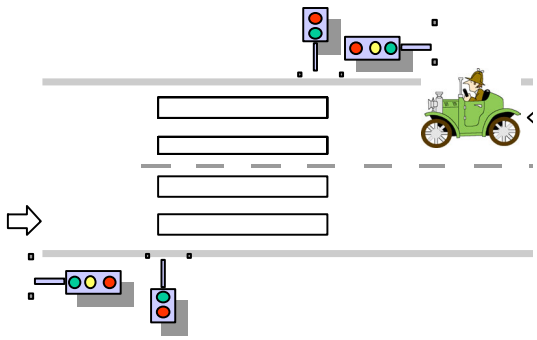
Slika 6.1-1. Blokovni prikaz.

V drugem delu naloge realizirajte števnik z dvomestnim desetiškim prikazovalnikom, ki prikazuje trenutno število parkiranih avtomobilov na parkirišču, na katerem je prostora za 50 vozil. Na vhodu v parkirišče je postavljen semafor. Ko je parkirišče polno, se zelena luč semaforja preklopi na rdečo in ostane rdeča tako dolgo, dokler najmanj eno od parkiranih vozil parkirišča ne zapusti. Omogočimo tudi nastavljanje in brisanje vrednosti v števniku, ki prikazuje število trenutno parkiranih vozil. Desetiški števnik s trimestnim desetiškim prikazovalnikom pa je namenjen štetju vseh parkiranih vozil za neko časovno obdobje (dan, teden, mesec ...).

6.2 Projekt 2: Semaforiziran prehod za pešce 1

Besedilo naloge

Realizirajte vezje za krmiljenje semaforjev prehoda za pešce (slika 6.2-1) na dvosmerni cesti. Pešci s pritiskom na gumb lahko opozorijo na svojo prisotnost oz. izrazijo željo po prečkanju ceste. Ko pešec pritisne na gumb (signal C), se prične postopek krmiljenja luči na semaforjih. Signal ST (˝set timer˝) v vezju sproži časovnik (˝timer˝), ki ima dva izhoda. Na izhodu časovnika se signal TS postavi na 1, ko poteče kratki časovni interval, in signal na izhodu TL se postavi na 1, ko poteče dolgi časovni interval. Ko krmilno vezje dobi informacijo, da je potekel dolgi časovni interval od trenutka, ko je signal C na 1, se zelena luč na semaforju za avtomobile (gori en dolgi časovni interval) preklopi na rumeno (gori en kratki časovni interval) in rdeče (gori en dolgi časovni interval). Nasproti temu pa se preklapljajo luči na semaforju za pešce. Ko gori zelena ali rumena luč na semaforju za avtomobile, gori rdeča na semaforju za pešce. Ko gori rdeča na semaforju za avtomobile, pa gori zelena na semaforju za pešce. Vhodne in izhodne spremenljivke lahko opišemo na naslednji način:



Slika 6.2-1. Semaforiziran prehod za pešce.

Oznaka vhodne spremenljivke in opis:	
reset	postavi krmilno vezje v začetno stanje,
C	signal za prečkanje ceste,
TS	poteče kratki časovni interval,
TL	poteče dolgi časovni interval.

Oznaka izhodne spremenljivke in opis:	
HG, HY, HR	zahteva vklop zelene, rumene ali rdeče luči na semaforju za avtomobile,
PG , PR	zahteva vklop zelene in rdeče luči na semaforju za pešce,
ST	nastavi začetek merjenja kratkega oz. dolgega časovnega intervala.

Stanja kontrolnega vezja in opis:	
S0	za avtomobile gori zelena luč (za pešce rdeča),
S1	za avtomobile gori rumena luč (za pešce rdeča),
S2	za avtomobile gori rdeča (za pešce gori zelena luč).

6.3 Projekt 3: Semaforiziran prehod za pešce 2

Besedilo naloge

Realizirajte vezje, ki bo krmililo semaforje na prehodu za pešce (slika 6.2-1). Semaforji delujejo v dnevnem ali nočnem načinu. Podnevi semaforji ponavljajo naslednje zaporedje. Najprej se na semaforju za avtomobile prižge zelena luč in gori en dolgi časovni interval, nato gori en kratki časovni interval rumena luč in rdeča en dolgi časovni interval. Hkrati pa se na semaforju za pešce preklapljajo luči v nasprotnem vrstnem redu. Ko gori zelena ali rumena luč na semaforju za avtomobile, gori rdeča na semaforju za pešce, ko pa gori rdeča na semaforju za avtomobile, gori zelena na semaforju za pešce. Postopek se ponavlja, dokler časovnik (signal C) ne javi nočnega načina delovanja. Tedaj na semaforju za avtomobile prične rumena luč utripati, na semaforju za pešce pa utripa zelena luč. Dolgi in kratki časovni interval ter preklop z dnevnega na nočni način realiziramo s časovnikom (˝timer˝), ki ima en vhod in tri izhode. Vhodni signal ST (˝set timer˝) rabi za proženje časovnika (˝timer˝). Izhodni signal časovnika TS se postavi na 1, ko poteče kratki časovni interval, signal izhoda TL pa se postavi na 1, ko poteče dolgi časovni interval. Ko je signal izhoda C na 1, se prične nočni način delovanja krmilnega vezja. Vhodne in izhodne spremenljivke lahko opišemo na naslednji način:

Oznaka vhodne spremenljivke in opis:	
reset	postavi kontrolno vezje v začetno stanje,
C	signal, ki določi nočni način delovanja,
TS	poteče kratki časovni interval,
TL	poteče dolgi časovni interval.

Oznaka izhodne spremenljivke in opis:	
HG, HY, HR	zahteva vklopa zelene, rumene ali rdeče luči na semaforju za avtomobile,
PG, PR	zahteva vklopa zelene in rdeče luči na semaforju za pešce,
ST	nastavi začetek merjenja kratkega oz. dolgega časovnega intervala.

Stanja kontrolnega vezja in opis:	
S0	za avtomobile gori zelena luč (za pešce rdeča),
S1	za avtomobile gori rumena luč (za pešce rdeča),
S2	za avtomobile gori rdeča (za pešce gori zelena luč),
S3	za avtomobile utripa rumena luč (za pešce utripa zelena luč).

6.4 Projekt 4: Semaforizirano križišče

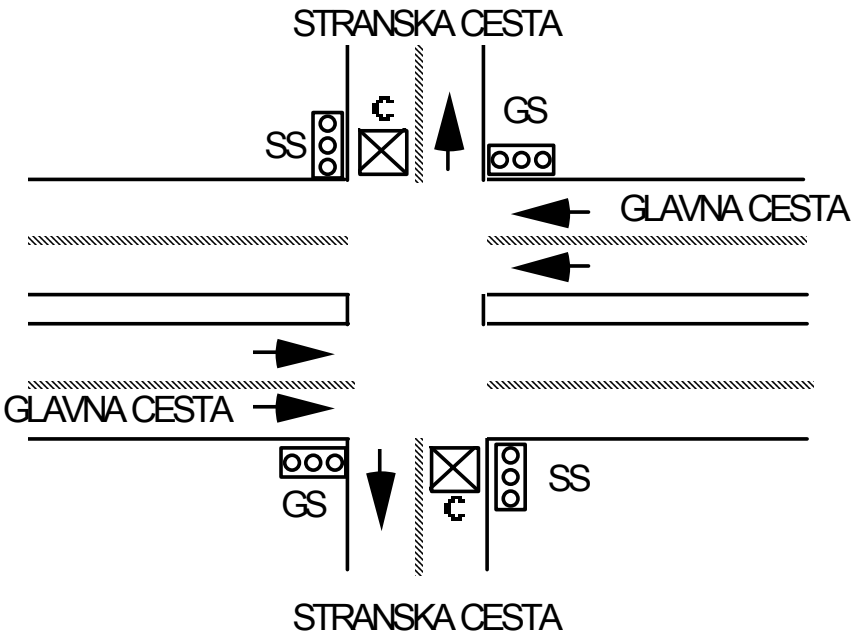
Besedilo naloge

Realizirajte vezje za semaforizacijo križišča dveh dvosmernih cest (glavne in stranske), kot je prikazano na sliki 6.4-1. Na stranski cesti je vgrajen induktivni detektor, ki na 1 postavi signal C takoj, ko zazna čakajoče vozilo. Preklopno vezje za krmiljenje semaforjev deluje na naslednji način. Zelena luč semaforjev na glavni cesti gori tako dolgo, dokler na stranski cesti ni vozila. Ko senzor zazna vozilo na stranski cesti, se luči na semaforjih glavne ceste preklopijo z zelene na rumeno in rdečo, medtem ko se na stranski cesti luči semaforjev preklopijo z rdeče na rumeno in zeleno. Zelena luč na semaforjih stranske ceste gori tako dolgo, dokler detektor javlja prisotnost vozil, vendar le do izteka enega časovnega intervala, ko lahko gori zelena luč na semaforjih stranske ceste. Ta časovna omejitev je zato, da zagotovimo tekoč promet na bolj obremenjeni glavni cesti. Ko so izpolnjeni pogoji, semaforja na stranski cesti preklopita z zelene na rumeno in nato rdečo luč, semaforja na glavni cesti pa z rdeče na rumeno in zeleno. Tudi če na stranski cesti še čakajo vozila, gori na semaforjih glavne ceste zelena luč, dokler ne poteče določen časovni interval. Zato realiziramo zunanji časovnik (˝timer˝) z izhodom TS, ki se postavi na 1 po kratkem časovnem intervalu (trajanje gorenja rumene luči na glavni cesti), in TL, ki se postavi na 1 po dolgem časovnem intervalu (trajanje gorenja rdeče oz. zelene luči na glavni cesti). Časovnik poženemo s krmilnim signalom ST (˝set timer˝). Vhodne in izhodne spremenljivke lahko opišemo na naslednji način:

Oznaka vhodne spremenljivke in opis:	
reset	krmilnik vrne v začetno stanje,
C	zazna čakajoče vozilo na obeh straneh stranske ceste,
TS	potekel je kratki časovni interval,
TL	potekel je dolgi časovni interval.

Oznaka izhodne spremenljivke in opis:	
HG, HY, HR	zahteva vklopa zelene, rumene ali rdeče luči na glavni cesti,
FG, FY, FR	zahteva vklopa zelene, rumene ali rdeče luči na stranski cesti,
ST	nastavi začetek merjenja kratkega oz. dolgega časovnega intervala.

Stanja kontrolnega vezja in opis:	
S0	na glavni cesti gori zelena luč (na stranski rdeča),
S1	na glavni gori rumena luč (na stranski rdeča),
S2	na stranski gori zelena luč (na glavni rdeča),
S3	na stranski gori rumena luč (na glavni rdeča).



Slika 6.4-1. Semaforizirano križišče.

6.5 Projekt 5: Vezje za krmiljenje dviganja/spuščanja avtomatske zapornice 1

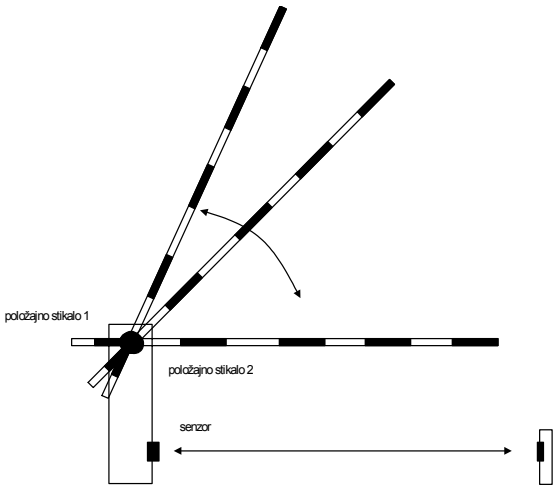
Besedilo naloge

Izdelajte vezje za krmiljenje dviganja, spuščanja in zaščite avtomatske zapornice. Signali za odpiranje, zapiranje in zaustavljanje zapornice so podani v tabeli 6.5-2.

Tabela 6.5-1. Kombinacije signalov, ki jih v vezje pošiljamo z daljinskim krmilnikom.

	A B
signal za dviganje zapornice	0 1
signal za spuščanje zapornice	1 0
signal za prekinitev dviganja/ spuščanja zapornice	1 1

Vezje ima dva vhoda, vhod A in B. Če je kombinacija vhodnih signalov 01, bo krmilnik določil dviganje zapornice, pri kombinaciji signalov 10 spuščanje, pri kombinaciji 11 pa zaustavljanje. Kombinacije vhodnih signalov vezja so realizirane z daljinskim upravljalnikom in jih vzemimo kot dejstvo. Zapornica ima tudi varnostni senzor, ki je nameščen v njenem območju. Signal senzorja ima na izhodu 0, ko ni ovire, pri oviri, ko je snop senzorja prekinjen, pa je izhodni signal senzorja na 1. V primeru, ko ovira v območju zapornice prekine snop senzorja, se mora dviganje ali spuščanje zapornice takoj prekiniti. Po odstranitvi ovire se postopek dviganja ali spuščanja zapornice nadaljuje. Zapornica ima vgrajeni položajni stikali za določanje skrajnega položaja zapornice (slika 6.5-1).



Slika 6.5-1. Sistem zapornice.

6.6 Projekt 6: Vezje za krmiljenje dviganja/spuščanja avtomatske zapornice 2

Besedilo naloge

Izdelajte vezje za krmiljenje dviganja, spuščanja in zaščite avtomatske zapornice. Signali za odpiranje, zapiranje in zaustavljanje zapornice so podani v tabeli 6.6-1.

Tabela 6.6-1. Zaporedja signalov, ki jih v vezje pošiljamo z daljinskim krmilnikom.

zaporedje za dviganje zapornice	1 0 1
zaporedje za prekinitev dviganja/ spuščanja zapornice	0 1 0

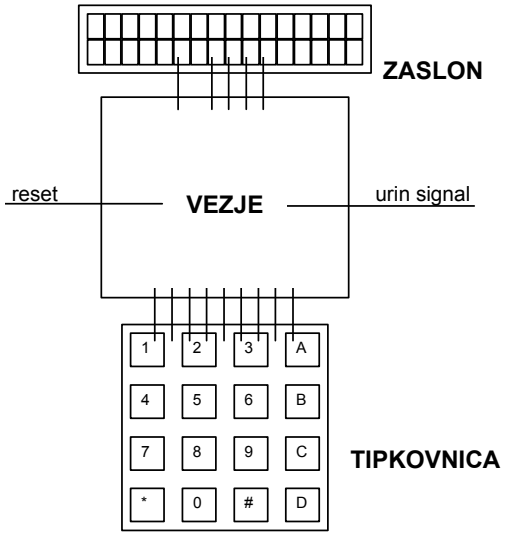
Vezje ima vhod, na katerem ugotavlja zaporedje vhodnih signalov. Če bo ugotovljeno zaporedje signalov 101, naj krmilnik zapornico dviga, s kombinacijo signalov 010 pa dviganje prekinemo. Zaporedja signalov na vhodu vezja so realizirana z daljinskim upravljalnikom. Zapornica ima tudi varnostni senzor, ki je nameščen v območju zapornice (slika 6.5-1).

Signal senzorja ima na izhodu 0, ko ni ovire, pri oviri, ko je snop senzorja prekinjen, pa je izhodni signal senzorja na 1. V primeru, ko ovira v območju zapornice prekine snop senzorja, se mora dviganje ali spuščanje zapornice takoj prekiniti. Ko je ovira odstranjena, se dviganje ali spuščanje zapornice nadaljuje. Zapornica ima vgrajeni položajni stikali za določanje skrajnega položaja zapornice. Zapornica ostane v položaju ‘odprto’ samo nekaj časa, ko ta preteče, pa se prične spuščati. Izdelajte števnik, ki bo prikazoval čas do trenutka, ko se bo zapornica pričela ponovno spuščati. Realizirajte vezje, s katerim bo mogoče določiti čas odprte zapornice.

6.7 Projekt 7: Prikazovanje izbranih telefonskih številk ter avtomatsko pozivanje

Besedilo naloge

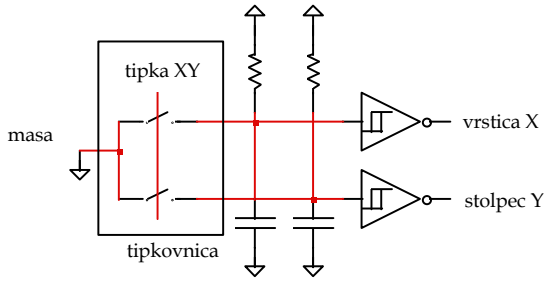
Izdelajte vezje za prikazovanje izbranih telefonskih številk na LCD prikazovalniku. Če v petih sekundah od vtipkanja zadnje telefonske številke ne pritisnemo nobene od tipk na telefonski tipkovnici, se prične postopek odpošiljanja signala za povezavo, ko se postopek konča, pa se zaslon zbriše in resetira. Blokovna shema projekta je prikazana na sliki 6.7-1. Realizirajte vezje, ki je sestavljeno iz treh funkcijskih blokov: tipkovnice, zaslona (LCD prikazovalnik) ter krmilnega vezja. LCD zaslon sestavite iz šestih prikazovalnikov šestnajstiške številke, ki imajo po štiri vhode. Če izhode tipkovnice povežemo neposredno s prikazovalnikom, ta pokaže vrednost aktivne tipke (aktivna tipka je tista, ki je bila nazadnje pritisnjena). Privzeta vrednost na vhodih je 0000 ali šestnajstiška 0, kot je to na izhodu tipkovnice. Krmilno logično vezje dobiva informacije z 8 vhodov s tipkovnice, z linije za resetiranje, ima pa 4 izhode za LCD prikazovalnik ter dodatnega za resetiranje prikazovalnika. Hitrost ure je poljubna, realiziramo jo sami.



Slika 6.7-1. Blokovna shema vezja telefonske tipkovnice in prikazovalnika.

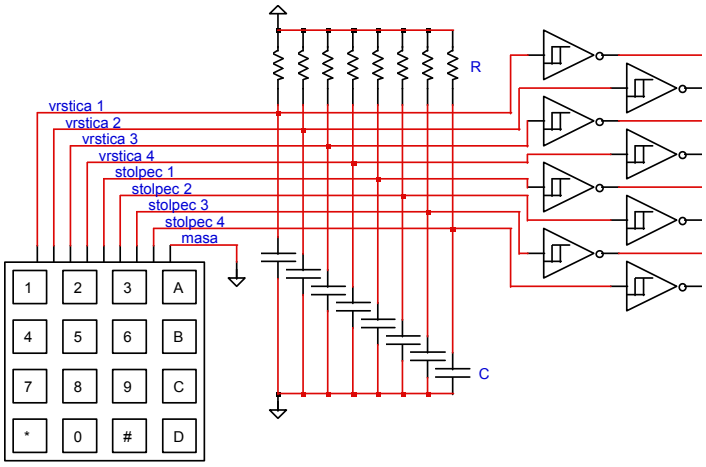
Tipkovnica:

Ima 9 izhodov. Za 16 tipk na tipkovnici (4x4) nam izhodi od 1 do 4 predstavljajo vrstice, izhodi od 5 do 8 pa predstavljajo stolpce, v katerih ležijo tipke (slika 6.7-2). Zadnji izhod (9) je skupna povezava, ki je povezana na maso.



Slika 6.7-2. Shema za posamezno tipko.

Ko pritisnemo eno od tipk, se izbrana vrstica in stolpec povežeta na maso, tako da je vrednost signala na izbranih povezavah na 0 (slika 6.7-3). Dva od osmih izhodov iz tipkovnice sta na 0, ostali pa so na 1 (+5V). Nezaželenemu preklapljanju signala ob preklopu tipke se izognemo z členom RC in integriranim vezjem vrste ″Schmitt trigger″ (Schmittovo prožilno vezje 74LS14). Vrednosti elementov R in C izberemo tako, da dobimo čas preklopa približno 50 ms.



Slika 6.7-3. Blokovna shema telefonske tastature.

Na izbrani povezavi je invertor, ki ima na vhodu Schmittovo prožilno vezje. Ko tipko sprostimo, se kondenzator prek uporov prične ponovno polniti in invertor po določenem času ponovno preklopi na 0. Realizirajte vezje, ki bo kodiralo pritisnjeno tipko v ustrezno binarno število (od 0000 do 1001 za števila tipk od 0 do 9, A, B, C in D lahko imajo poljubne vrednosti X, število 1010 je koda za *, 1111 pa koda za #). Peta linija je na 0, ko je pritisnjena ena izmed tipk, in rabi za sinhronizacijo z uro. Zagotovite, da bo ta linija aktivna vedno takrat, ko bo na preostalih štirih linijah veljavna koda.

6.8 Projekt 8: Preprosta varnostna naprava

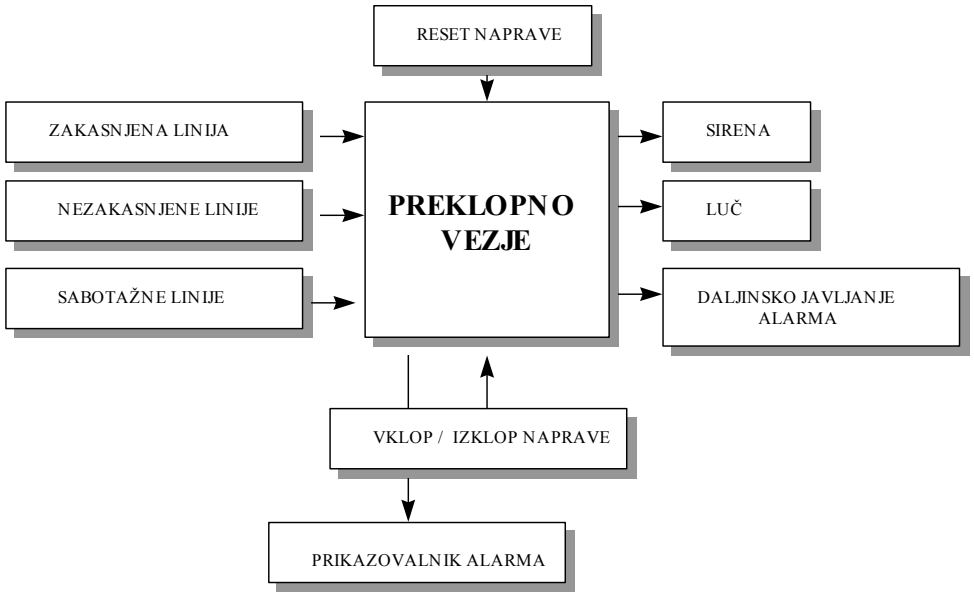
Besedilo naloge

Sestavite vezje za preprosto alarmno napravo (slika 6.8-1). Naprava ima funkcijo za vklop in izklop (on/off), zakasnjeno, direktno in sabotažno linijo, zvočni in svetlobni signal, indikator za pomnjenje alarma ter stikalo za resetiranje vezja.

Splošna navodila za izdelavo vezja:

Alarmno napravo sestavljajo naslednji funkcijski elementi:

- preklopno-krmilno vezje,
- zvočna signalizacija alarma,
- svetlobna signalizacija alarma,
- daljinsko javljanje alarma,
- vklop/izklop naprave,
- resetiranje naprave,
- zakasnjena linija,
- nezakasnjene linije,
- sabotažne linije,
- prikazovalnik alarma.



Slika 6.8-1. Blokovna shema alarmne naprave.

Preklopno-krmilno vezje

je osrčje naprave. Nadzira vhodne linije in določa signale izhodnih linij.

Zvočna signalizacija alarma

Pri aktiviranju alarma se vklopi zvočni signal (sirena), ki ostane vklopljen v nekem časovnem intervalu. Realizirajte možnost nastavitve časovnega intervala (od 1 do 8 časovnih enot). Pri ponovnem aktiviranju senzorjev na alarmnih linijah se zvočni signal ponovno aktivira.

Svetlobna signalizacija alarma

Hkrati z aktiviranjem zvočnega alarma se aktivira svetlobni alarm. Svetlobni signal se izklopi ob ‘razostritvi’ alarmne naprave ali ob resetiranju naprave.

Vklop in izklop naprave

je mogoč preko tipkovnice ali ključavnice. Rabi za vsakodnevno uporabo.

Sabotažne linije

so linije, ki so ves čas ‘ostre’. Alarm se aktivira brezpogojno in takoj, neglede na to, ali je alarmna naprava vključena ali izključena (‘ostra’ ali ne). Sabotažne linije ščitijo samo napravo, alarmne linije in senzorje pred namernim uničenjem.

Nezakasnjene linije

Alarm se aktivira takoj, ko nastane prekinitev na liniji, pri pogoju, da je naprava vklopljena (‘ostra’). Ob vsaki nadaljnji prekinitvi nezakasnjenih linij se alarm ponovno aktivira.

Zakasnjena linija

Uporabimo jo v prostoru, kjer vklapljamo ali izklapljamo napravo. Časovni interval od vklopa do ‘naostritve’ naprave ter od aktiviranja zakasnjene linije pri prihodu do trenutka aktiviranja alarma je nastavljiv (od 1 do 8 časovnih enot). Ponavadi zadošča samo ena zakasnjena linija.

Prikazovalnik alarma

Prikazovalnik alarma prikazuje stanje oz. zgodovino alarmov. Uporabniku da informacijo, da je med njegovo odsotnostjo prišlo do aktiviranja alarma. Prikazovalnik ostane aktiven tako dolgo, dokler naprave ponovno ne vklopimo. Boljše izvedbe imajo tudi števnik alarmov.

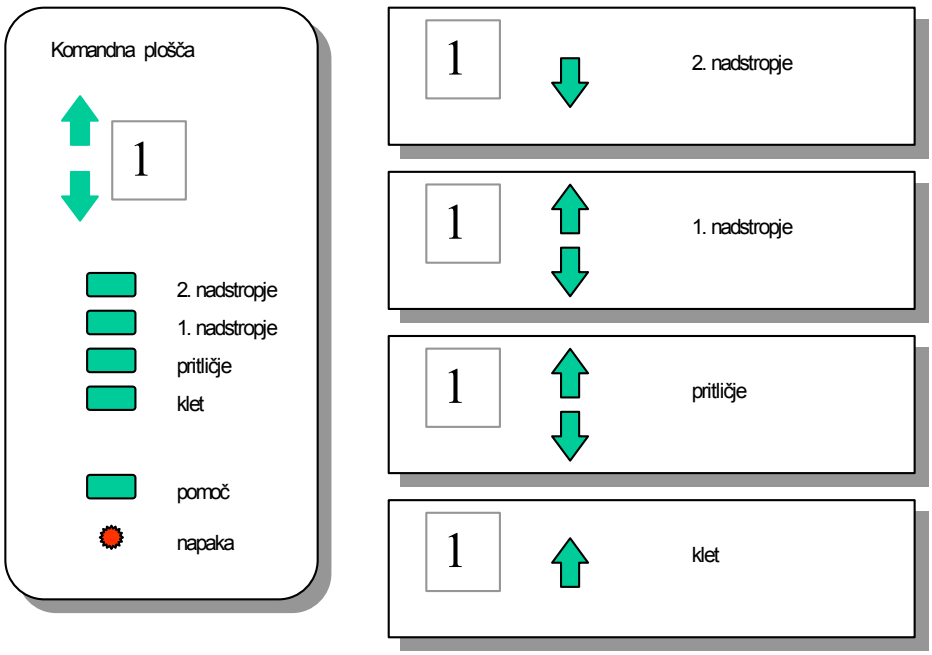
Daljinsko javljanje alarma

Sproži se signal za prikrit alarm, ki prikliče intervencijsko službo.

6.9 Projekt 9: Dvigalo za prevoz oseb

Besedilo naloge

Realizirajte vezje za krmiljenje dvigala za prevoz ljudi v dvonadstropnem stanovanjskem objektu s kletjo. V dvigalu je nameščena komandna plošča. S tipkami izbiramo nadstropje, v katero se želimo pripeljati. V dvigalu je nameščeno stikalo za klic v sili v primeru okvare, signalizacija za preobremenjenost dvigala ter digitalni prikazovalnik za informacijo, v katerem nadstropju se nahajamo. Na vhodih v posameznih nadstropjih so nameščene tipke za klicanje dvigala ter prikazovalnik nadstropja, v katerem se dvigalo nahaja (slika 6.9-1). Vezje sproti ugotavlja in upošteva smiselne zahteve potnikov, ki čakajo na dvigalo, na poti gibanja dvigala.



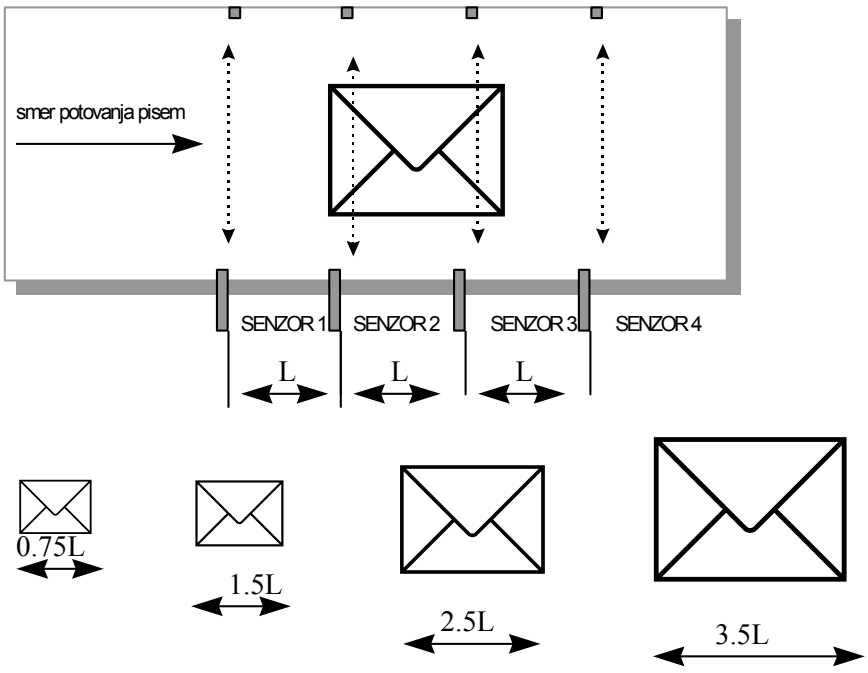
Slika 6.9-1. Blokovna shema dvigala.

V kolikor se smer gibanja dvigala in zahtevana smer potnika, ki na dvigalo čaka, ujemata, se dvigalo ustavi, v kolikor ne, pa se vrne po potnika potem, ko opravi prvotno nalogo. V ta namen izdelajte pomnilnik FIFO, v katerega se vpisuje seznam zahtevkov.

6.10 Projekt 10: Pisemski razvrščevalnik

Besedilo naloge

Realizirajte vezje za krmiljenje razvrščevalnika pisem po njihovi velikosti. Pisma potujejo po tekočem traku, na katerem so nameščeni optični senzorji, med seboj oddaljeni za razdaljo L (slika 6.10-1). Upoštevamo, da po traku potujejo le pisemske ovojnice standardnih dolžin 0.75 L, 1.5L , 2.5L in 3.5L. Vezje ima tudi števnike, ki prikazujejo število pisem, razvrščenih po velikosti.

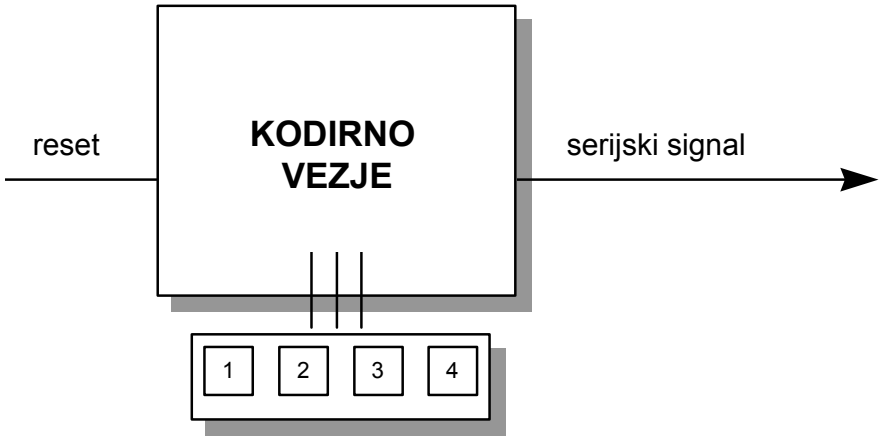


Slika 6.10-1. Razvrščevalnik za pisma.

6.11 Projekt 11: Daljinski upravljalnik

Besedilo naloge

Realizirajte vezje za daljinski upravljalnik, ki lahko kodira 4 funkcije (slika 6.11-1). Kodirnik izdelajte tako, da bo pretvarjal paralelne signale, ki jih dobimo iz tipkovnice, v serijski signal. Kodiran serijski signal opremite z zaporedjem signalov, ki označujejo začetek in konec pošiljanja informacijskega signala.

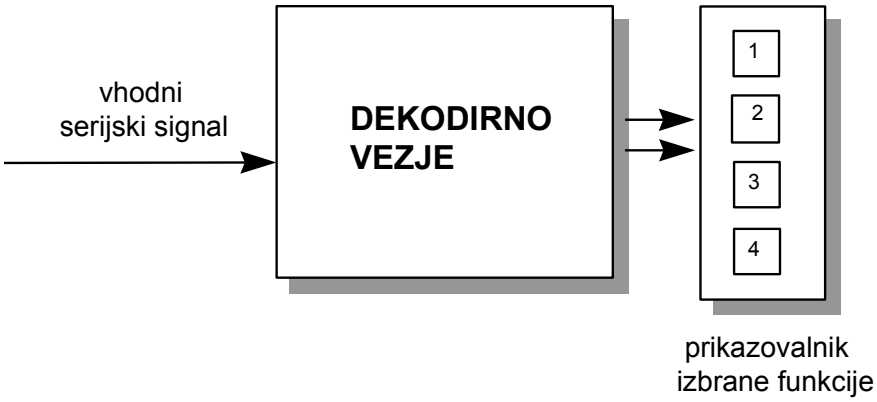


Slika 6.11-1. Daljinski upravljalnik.

6.12 Projekt 12: Sprejemnik in dekodirnik signala

Besedilo naloge

Realizirajte vezje sprejemnika serijskega signala, ki bo dekodiralo 4 različne vhodne kombinacije. Vezje ima en vhod, na katerega pride kodiran daljinski serijski signal (slika 6.12-1). Izdelajte dekodirno vezje ter serijsko/paralelni pretvornik za sprejemanje signalov iz daljave. Izdelajte krmilno vezje, s katerim bo mogoče prožiti štiri različne funkcije sprejemnika.

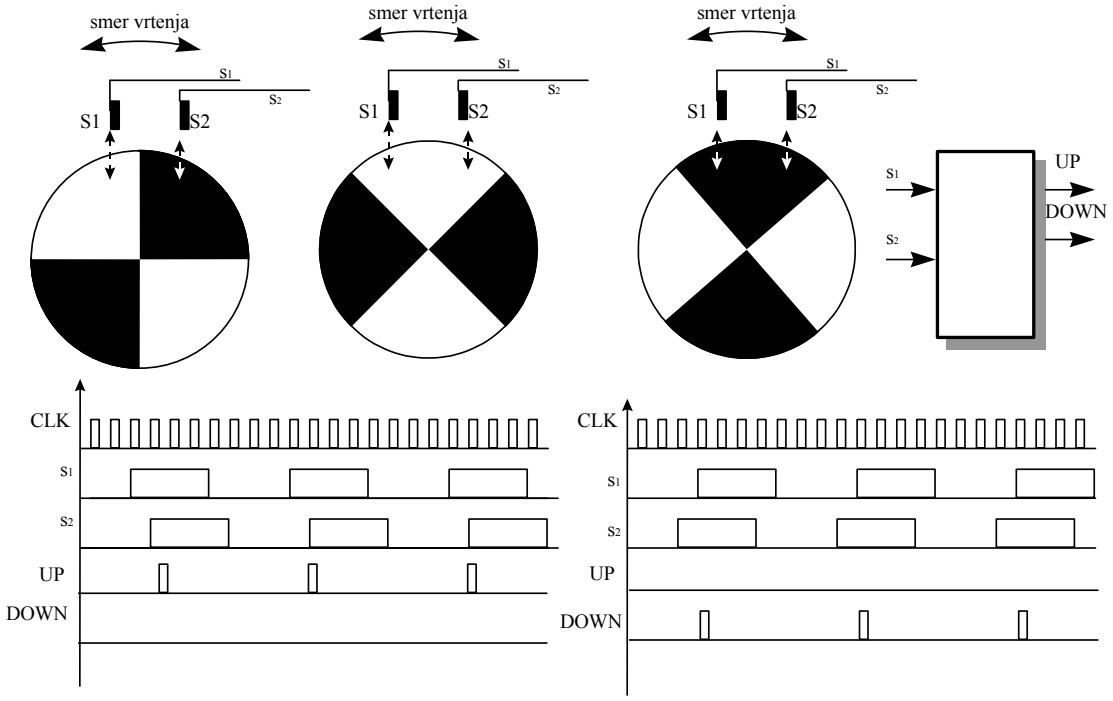


Slika 6.12-1. Sprejemno-dekodirno vezje.

6.13 Projekt 13: Inkrementalni dajalnik

Besedilo naloge

Realizirajte vezje za ugotavljanje smeri vrtenja osi, na kateri je nameščen inkrementalni dajalnik (slika 6.13-1). Dodajte vezje za štetje impulzov oz. za določanje pozicije.



Slika 6.13-1. Inkrementalni dajalnik s pripadajočimi vhodnimi in izhodnimi signali.

7 REŠITVE PROJEKTOV S PROGRAMSKIM PAKETOM LOGICWORKS

UVOD

V nadaljevanju podajamo rešitve nekaterih projektov iz prejšnjega poglavja, kot so si jih zamislili nekateri študentje. Večina študentov se je pri izdelavi projekta zelo potrudila in rešitve tudi odlično predstavila. Prikazali bomo dopolnjene in delno preurejene rešitve nekaterih projektov. Primer projekta Semaforizirano križišče pa je podan v angleškem jeziku, realiziran pa je bil v laboratoriju za elektroniko na univerzi Berkeley. Z zgledi in idejami želimo tako pomagati študentom pri izdelavi njihovih projektov. Ker v posameznem letniku več študentov rešuje isti projekt, je veljal dogovor, da včasih ni potrebno upoštevati stroge specifikacije zahtev pri posameznem projektu, temveč so dopustne tudi nekatere spremembe oz. odstopanja od originalnih zahtev. Zaradi tega se je povečala kreativnost pri delu, rezultat pa so tudi bolj zanimive in predvsem raznolike rešitve.

SEZNAM PROJEKTOV

Projekt 1: Avtomatizacija parkirišča

Projekt 2: Semaforiziran prehod za pešce 1

Projekt 3: Semaforiziran prehod za pešce 2

Projekt 4: Semaforizirano križišče

Projekt 5: Vezje za krmiljenje dviganja/spuščanja avtomatske zapornice 1

Projekt 7: Prikazovanje izbranih telefonskih številk ter avtomatsko pozivanje

Projekt 8: Preprosta varnostna naprava

Projekt 9: Dvigalo za prevoz oseb

Projekt 12: Sprejemnik in dekodirnik signala

Projekt 13: Inkrementalni dajalnik

7.1 Projekt 1: Avtomatizacija parkirišča

Projekt razdelimo na naslednje podsklope:

- sinteza vezja, ki ugotavlja smer prečkanja kontrolnega mesta,
- sinteza števnikov, ki prikazujeta trenutno in skupno število parkiranih vozil v izbranem obdobju,
- sinteza vezja za krmiljenje semaforja (parkirišče je zasedeno/prosto),
- sinteza vezja za nastavljanje (set) in brisanje (reset) dnevnega števnika.

Opis delovanja vezja

V nadaljevanju opišemo vezje, ki je sestavljeno iz štirih sklopov:

- vezja za ugotavljanje smeri prečkanja kontrolnega mesta (v nadaljevanju vezje A),
- vezja, ki šteje trenutno parkirana vozila na parkirišču (v nadaljevanju vezje B),
- vezja, ki šteje skupno število parkiranih vozil v nekem obdobju (v nadaljevanju vezje C),
- vezja za krmiljenje semaforja (v nadaljevanju vezje D).

Vezje za ugotavljanje smeri prečkanja kontrolnega mesta:

a) Vhodne in izhodne spremenljivke

Vhodne spremenljivke: Izhodne spremenljivke:

x_1 – senzor 1 z_1

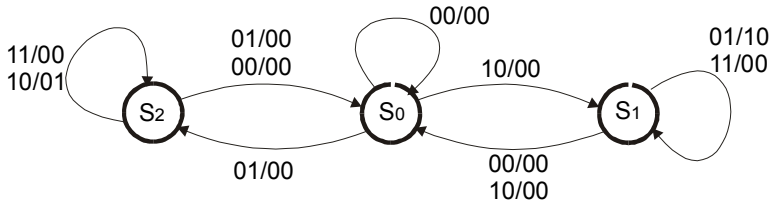
x_2 – senzor 2 z_2

b) Opis stanj

Stanje	y_1y_2	Opis
S_0	0 0	avtomobila ni
S_1	1 0	avtomobil je odpeljal s parkirišča
S_2	0 1	avtomobil je pripeljal na parkirišče

c) Diagram prehajanja stanj

x_1x_2/z_1z_2



Slika 7.1-1. Diagram prehajanja stanj, ki določa smer prehoda vozila.

d) Tabela prehajanja stanj

Sedanje stanje				Naslednst./izhodi				Pomnilni vhodi	
x_1	x_2	y_1	y_2	y_1^+	y_2^+	z_1	z_2	D_1	D_2
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	1	X	X	X	X	X	X
0	1	0	0	0	1	0	0	0	1
0	1	0	1	0	0	0	0	0	0
0	1	1	0	1	0	1	0	1	0
0	1	1	1	X	X	X	X	X	X
1	0	0	0	1	0	0	0	1	0
1	0	0	1	0	1	0	1	0	1
1	0	1	0	0	0	0	0	0	0
1	0	1	1	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X
1	1	0	1	0	1	0	0	0	1
1	1	1	0	1	0	0	0	1	0
1	1	1	1	X	X	X	X	X	X

D_1:

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	X	1
01	0	0	0	0
11	X	X	X	X
10	0	1	1	0

D_2:

y_1y_2 \ x_1x_2	00	01	11	10
00	0	1	X	0
01	0	0	1	1
11	X	X	X	X
10	0	0	0	0

z_1:

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	X	0
01	0	0	0	0
11	X	X	X	X
10	0	1	0	0

z_2:

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	X	0
01	0	0	0	1
11	X	X	X	X
10	0	0	0	0

e) Izhodne enačbe

$$D_1 = x_2y_1 + x_1\bar{y}_1\bar{y}_2$$

$$D_2 = x_1y_2 + x_2\bar{y}_1\bar{y}_2$$

$$z_1 = \bar{x}_1x_2y_1$$

$$z_2 = x_1\bar{x}_2y_2$$

Z vezjem A realiziramo diagram prehajanja stanj s tremi stanji (slika 7.1-1). Za vezje A (slika 7.1-2) potrebujemo dve pomnilni celici D in nekaj standardnih elementov logičnih vrat. Vrednosti vhodnih spremenljivk za vezje A dobimo s senzorjev X_1 in X_2. Kombinacija izhodov vezja A nam da informacijo o tem, ali je vozilo na parkirišče pripeljalo ali s parkirišča odpeljalo. Izhode z_1 in z_2 vezja A določajo vrednosti spremenljivk stanj y_1y_2 (00, 10 oz. 01) ter vrednosti spremenljivk na vhodih x_1 in x_2.

Vezji za štetje in prikaz trenutnega in skupnega števila parkiranih vozil

Vhodni del vezja B (slika 7.1-3) je kodirna logika, ki prekodira kombinacije izhodov z_1z_2 vezja A. Za štetje števila trenutno parkiranih vozil potrebujemo dva števnika gor/dol (vezje 4029 omogoča štetje v binarnem ali decimalnem zapisu). Števnik šteje gor, če je vhodni priključek U/D na 1, in dol, kadar je vhodni priključek U/D na 0. Za prikaz števila trenutno parkiranih vozil uporabljamo 7- segmentne LED prikazovalnike. Za dekodiranje koda BCD za 7-segmentne LED prikazovalnike uporabljamo dekodirno vezje 7449.

Vezje C (slika 7.1-4) je izvedeno podobno kot vezje B, le da nima na začetku prekodirnega vezja. Števnik v tem primeru šteje samo navzgor. Ker z vezjem C štejemo vsa vozila, ki so uporabila parkirišče v določenem časovnem obdobju (dan, mesec, leto), ima tri števnike 4029. Tudi za prikaz skupnega števila parkiranih vozil uporabljamo 7-segmentne LED prikazovalnike in dekodirna vezja 7449.

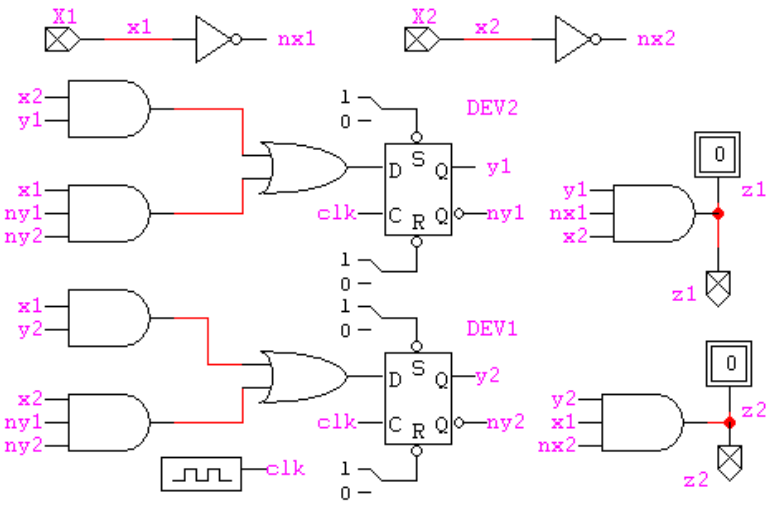
Krmilnik semaforja

Vezje D (slika 7.1-5) je preprosto krmilno vezje, ki krmili semafor. Zelena luč semaforja sveti, ko je parkirišče prosto (manj kot 50 parkiranih vozil), rdeča pa, ko je parkirišče zasedeno (na parkirišču je 50 vozil ali več).

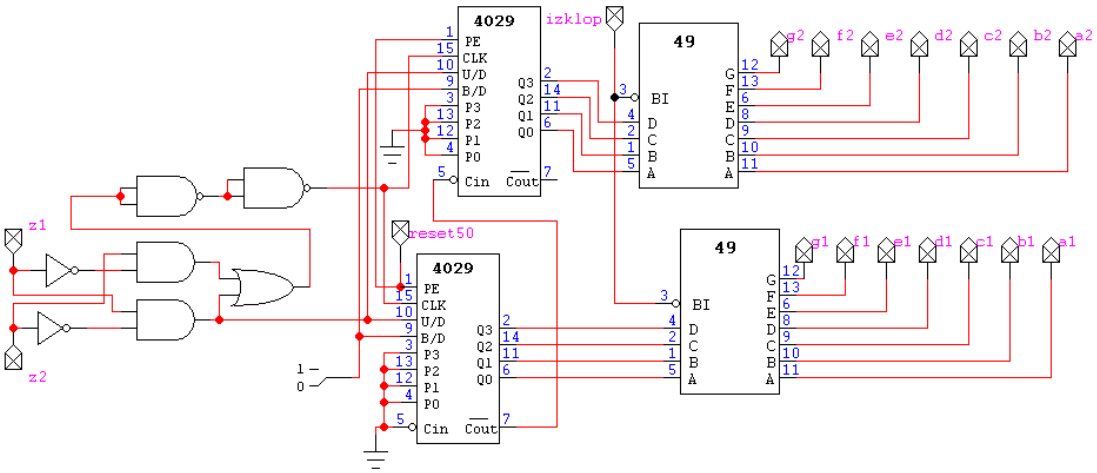
Celotno vezje

Celotno vezje (slika 7.1-6) je sestavljeno iz vseh do sedaj opisanih vezij. Tipke RESET so namenjene za brisanje posameznih števnikov. Število parkiranih vozil lahko nastavimo tudi ročno s stikali +1 oz. -1. Prihod in odhod vozil na parkirišče pa nastavimo ročno s stikali SENZOR1 in SENZOR2.

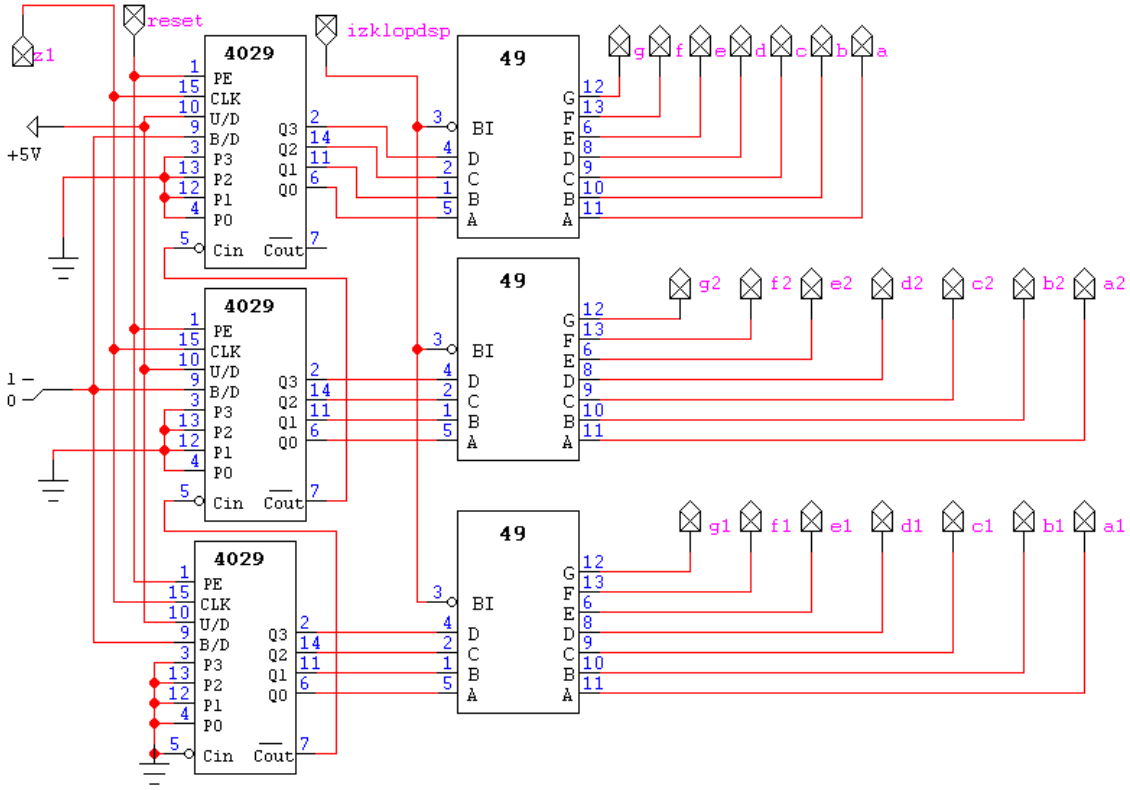
Vezalne sheme posameznih podvezij



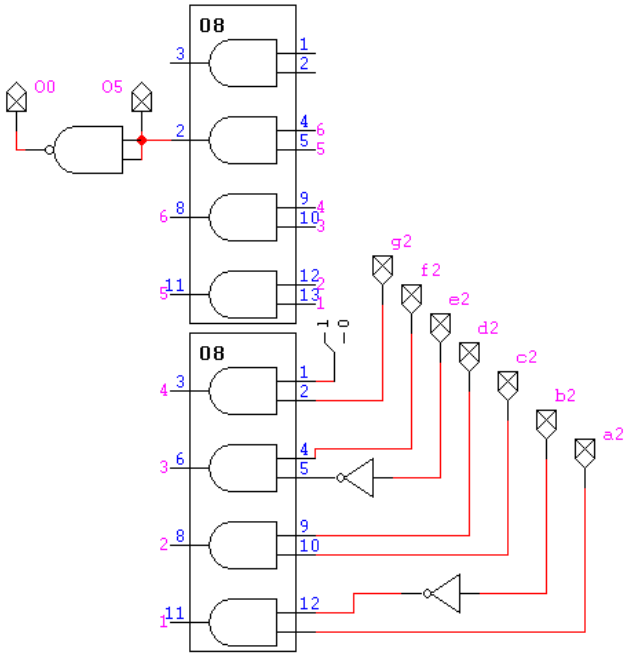
Slika 7.1-2. Vezje A.



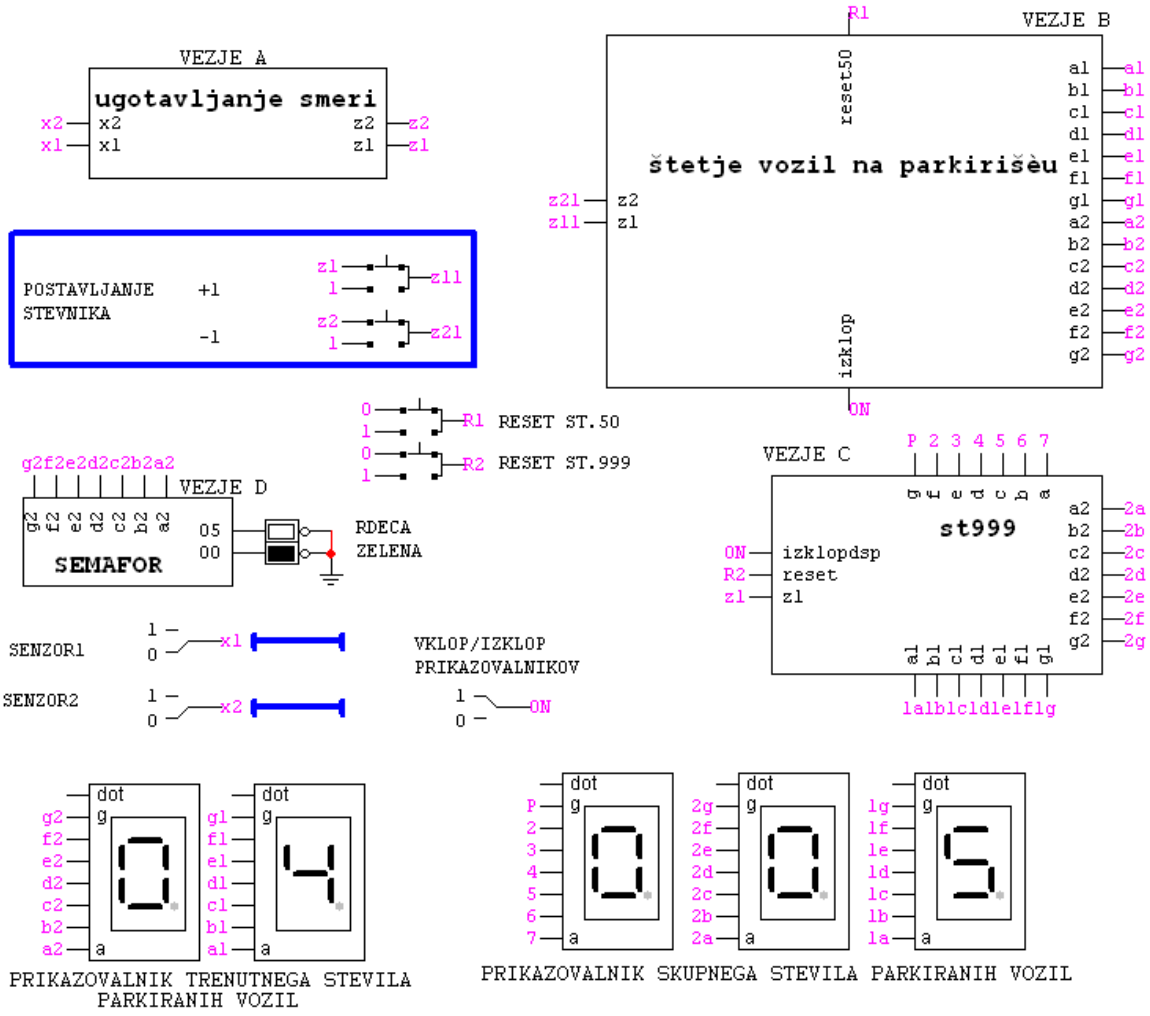
Slika 7.1-3. Vezje B.



Slika 7.1-4. Vezje C.



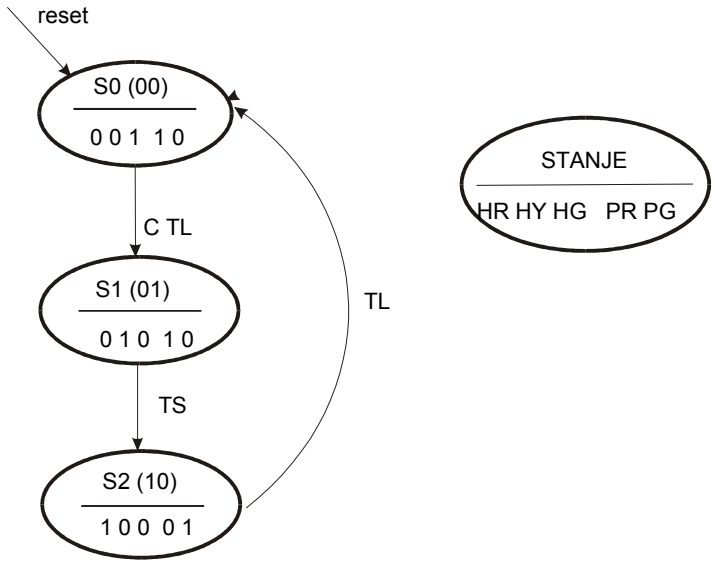
Slika 7.1-5. Vezje D.



Slika 7.1-6. Celotno vezje projekta Avtomatizacija parkirišča.

7.2 Projekt 2: Semaforiziran prehod za pešce 1

Realizirali smo vezje za krmiljenje semaforjev prehoda za pešce na dvosmerni cesti. Pešci s pritiskom na gumb lahko opozorijo na svojo prisotnost oz. izrazijo željo po prečkanju ceste. Ko pešec pritisne na gumb (signal C), se prične postopek krmiljenja luči na semaforjih. Signal C' v vezju sproži časovnik (˝timer˝), ki ima dva izhoda. Na izhodu časovnika se signal TS postavi na 1, ko poteče kratki časovni interval, in signal na izhodu TL se postavi na 1, ko poteče dolgi časovni interval. Ko krmilno vezje dobi informacijo, da je potekel dolgi časovni interval od trenutka, ko je signal C na 1, se zelena luč na semaforju za avtomobile (gori en dolgi časovni interval) preklopi na rumeno (gori en kratki časovni interval) in rdeče (gori en dolgi časovni interval). Nasproti temu pa se preklapljajo luči na semaforju za pešce. Ko gori zelena ali rumena luč na semaforju za avtomobile, gori rdeča na semaforju za pešce. Ko gori rdeča na semaforju za avtomobile, pa gori zelena na semaforju za pešce. Realizirali smo sekvenčno vezje tipa Moore, ki ima v vsakem stanju določeno kombinacijo izhodov. Opisane zahteve predstavimo z diagramom prehajanja stanj (slika 7.2-1). V diagramu so prikazane samo tiste vhodne kombinacije, ki povzročijo prehod v drugo stanje. Za realizacijo sekvenčnega vezja potrebujemo tri stanja, S0, S1 in S2. V vsakem stanju kombinacija signalov HR, HY in HG krmili luči semaforjev za avtomobile ter signala PR in PG luči semaforjev za pešce.



Slika 7.2-1. Diagram prehajanja stanj.

Diagram prehajanja stanj pretvorimo v tabelo prehajanja stanj (tabela 7.2-1). Tabelo smo v celoti realizirali z vezjem PROM po postopku, ki je opisan v poglavju 5.7.2. Dobili smo vezje "pesci_1c" (slika 7.2-2), ki ima pet vhodov in osem izhodov. Na vhode pripeljemo

spremenljivke y1, y2, C, TL in TS, na izhodih pa dobimo signale HR, HY in HG, ki krmilijo luči semaforjev za avtomobile, signala PR in PG, ki krmilita luči semaforjev za pešce, ter krmilna signala za pomnilni celici d1 in d2. S signalom ST postavimo časovnik v začetno stanje. Signal ST se postavi na 1 pri vsakem prehodu iz enega v drugo stanje.

Tabela 7.2-1. Tabela prehajanja stanj.

y_1	y_2	C	TL	TS	y^+_1	y^+_2	HR	HY	HG	PR	PG
0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	1	0	0	0	0	1	1	0
0	0	1	1	0	0	1	0	0	1	1	0
0	0	1	1	1	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0	1	0	1	0
0	1	0	1	0	1	0	0	1	0	1	0
0	1	0	1	1	1	0	0	1	0	1	0
0	1	1	0	0	0	1	0	1	0	1	0
0	1	1	0	1	1	0	0	1	0	1	0
0	1	1	1	0	1	0	0	1	0	1	0
0	1	1	1	1	1	0	0	1	0	1	0
1	0	0	0	0	1	0	1	0	0	0	1
1	0	0	0	1	1	0	1	0	0	0	1
1	0	0	1	0	0	0	1	0	0	0	1
1	0	0	1	1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	1	0	0	0	1
1	0	1	0	1	1	0	1	0	0	0	1
1	0	1	1	0	0	0	1	0	0	0	1
1	0	1	1	1	0	0	1	0	0	0	1
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X

Generator časovnih intervalov

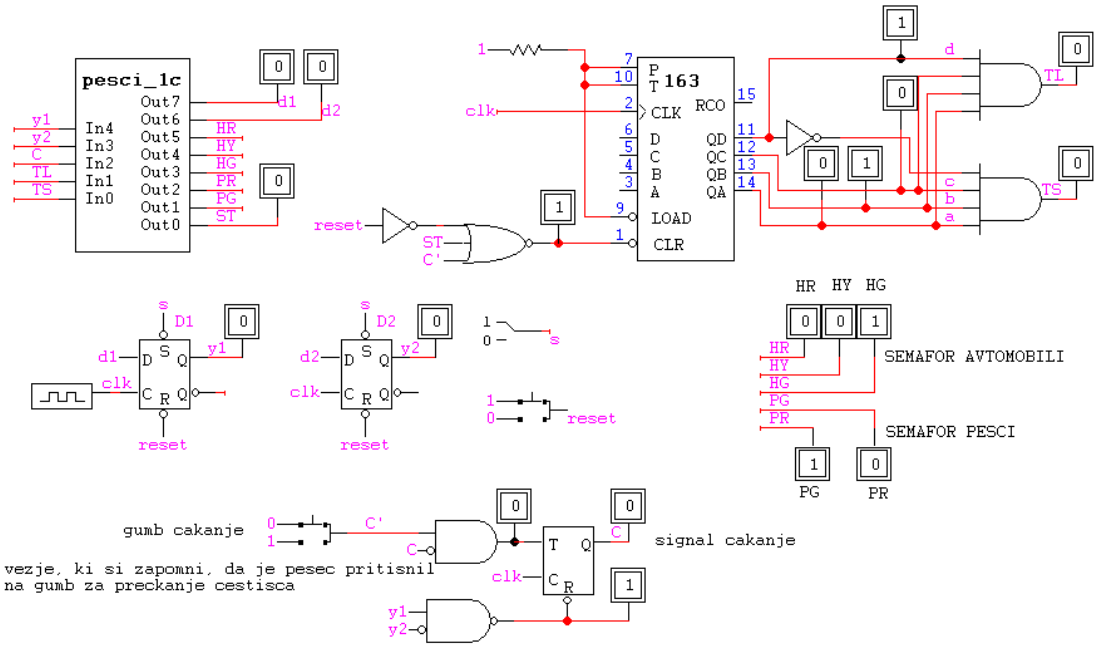
To je vezje, s katerim generiramo kratki (TS) in dolgi (TL) časovni interval. Načinov za realizacijo vezja je več. Mi smo se odločili, da vezje realiziramo s pomočjo binarnega števnika LS163. Dolžino kratkega (TS) in dolgega (TL) časovnega intervala smo določili s kombinacijo izhodnih bitov števnika QA-QD. Ko števnik prešteje osem bitov, poteče kratki časovni interval in signal TS gre na 1. Po šestnajstih bitih poteče dolgi časovni interval in na 1 gre signal TL. Pričetek štetja nastavimo z brisanjem števnika na priključku CLR (CLR postavimo na 0). Pogoj za brisanje števnika (CLR na 0) je izpolnjen, če je vsaj eden od signalov C´, ST ali ´reset´ na 1 (slika 7.2-2).

Signal za čakanje

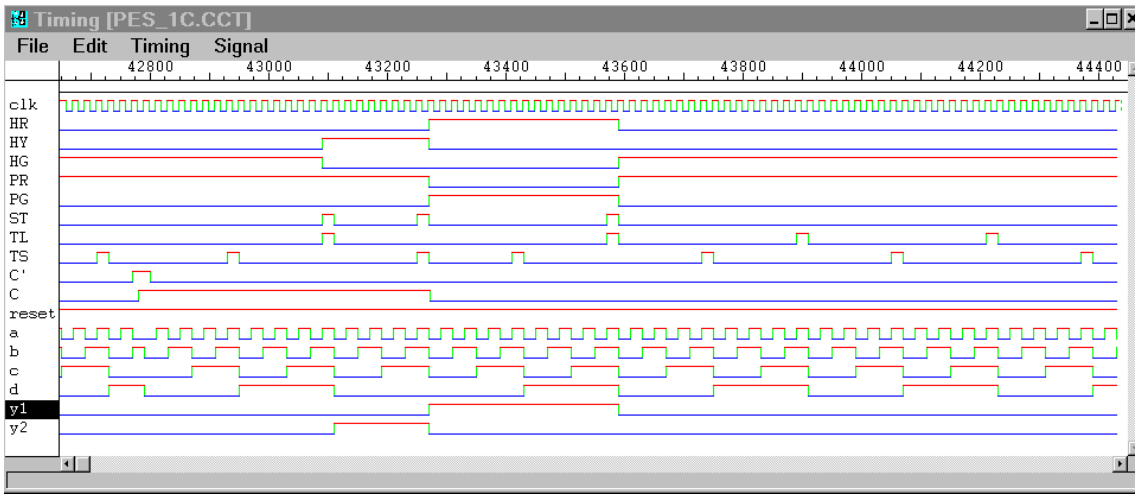
Realizirali smo tudi vezje, ki si zapomni, da je pešec pritisnil na gumb za čakanje. S pritiskom na gumb za čakanje (C' je na 1) postavimo signal za čakanje C na 1. Signal C ostane na 1 tako dolgo, dokler ni izpolnjen pogoj za prečkanje cestišča (za pešce mora goreti zelena luč). Z izpolnjenim pogojem postavimo signal C ponovno na 0.

Vezje

Celotno vezje je podano na sliki 7.2-2 , rezultat simulacije pa na sliki 7.2-3.



Slika 7.2-2. Vezje za semaforiziran prehod za pešce 1.



Slika 7.2-3. Prikaz simulacije vezja.

7.3 Projekt 3: Semaforiziran prehod za pešce 2

Realizirali smo vezje, ki krmili semaforje na prehodu za pešce. Semaforji delujejo v dnevnem ali nočnem načinu. Podnevi se semaforji preklapljajo po naslednjem zaporedju. Najprej se na semaforju za avtomobile prižge zelena luč in gori en dolgi časovni interval, nato gori en kratki časovni interval rumena luč in rdeča en dolgi časovni interval. Hkrati pa se na semaforju za pešce preklaplja luč z rdeče na zeleno in obratno. Vezje je sestavljeno iz naslednjih gradnikov: sekvenčnega vezja s pripadajočim krmilnim vezjem PESCI_2B, generatorja časovnih intervalov in vezja, ki določa čas trajanja nočnega načina delovanja celotnega vezja. V nadaljevanju podrobneje opišemo posamezne gradnike, s katerimi smo realizirali vezje.

Generator časovnih intervalov

To je vezje, s katerim generiramo kratki (TS) in dolgi (TL) časovni interval. Načinov za realizacijo vezja je več. Mi smo se odločili, da vezje realiziramo s pomočjo binarnega števnika LS163. Dolžino kratkega (TS) in dolgega (TL) časovnega intervala smo določili s kombinacijo izhodnih bitov števnika QA-QD. Ko števnik prešteje osem bitov, poteče kratki časovni interval in signal TS gre na 1. Po šestnajstih bitih poteče dolgi časovni interval in na 1 gre signal TL. Pričetek štetja nastavimo z brisanjem števnika na priključku CLR (CLR postavimo na 0). Pogoj za brisanje števnika (CLR na 0) je izpolnjen, če je vsaj eden od signalov C´, ST ali ´reset´ na 1 (slika 7.3-2). Signal C´ je v vezju namenjen izključno za ročno brisanje števnika.

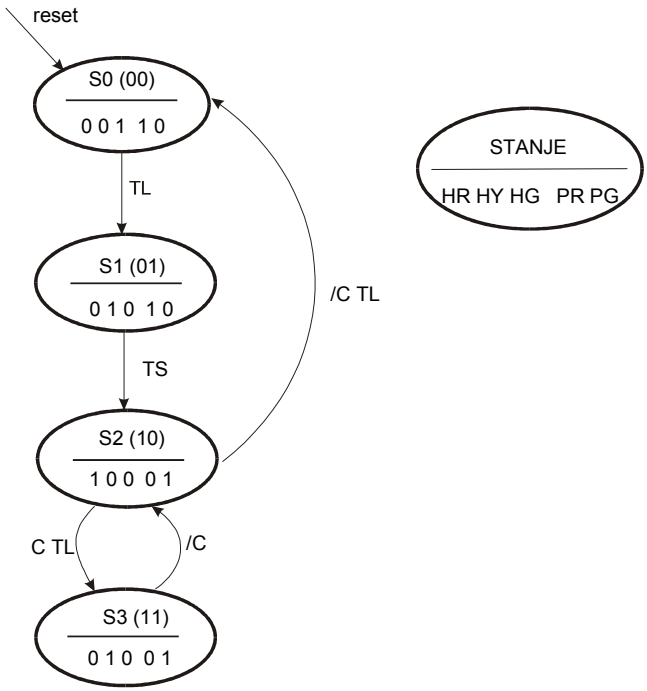
Vezje za generiranje signala za nočno-dnevni način delovanja

Realizirali smo preprosto vezje, ki določa način delovanja semaforjev. Ko signal C postane enak 1, preklopi vezje za krmiljenje semaforjev v nočni način delovanja. Vezje je sestavljeno iz preprostega paraleno/serijskega pomikalnega registra (slika 7.3-2). S signalom reset postavimo priključek LD na 0 in s tem register v paralelni način delovanja. Ob naslednjem urinem impulzu se prenesejo vrednosti z vhodnih priključkov D0-D7 (vsi priključki so na 0) na izhodne priključke Q0-Q7. Tako register postavimo v začetno stanje. V serijskem načinu register pri vsakem urinem impulzu prenese vrednost s priključka SI (SI je na 1) na izhod Q0, vrednost priključka Q0 na Q1 in tako naprej. Ko se postavi vrednost priključka Q7 na 1, se register postavi v paralelni način in ponovno prenese vrednosti z vhodnih priključkov D0-D7 na izhodne priključke Q0-Q7 ter se tako postavi ponovno v začetno stanje. Čas trajanja noči

lahko prilagodimo tako, da izberemo med enim izmed izhodnih signalov od Q2 (noc1) do Q6 (noc5). Signal noc1 predstavlja dolgo noč, signal noc5 pa kratko. Zaradi enostavnejšega testiranja smo dodali še ročni način preklopa dnevnega v nočni način in obratno. Na urin priključek registra CLK priključimo signal TL in tako primerno upočasnimo preklope.

Diagram prehajanja stanj

Delovanje vezja lahko predstavimo z diagramom prehajanja stanj (slika 7.3-1), ki ima štiri stanja. Vezje lahko s signalom reset v vsakem trenutku postavimo v začetno stanje S0. Pogoj za prehod iz stanja S0 v stanje S1 je, da poteče dolgi časovni interval (signal TL je na 1). Ko preteče kratki časovni interval (signal TS je na 1), je izpolnjen pogoj za prehod iz stanja S1 v stanje S2. V stanju S2 preverjamo vrednost signala C. Če je signal C na 0, ostanemo v dnevnem načinu in se po poteku dolgega časovnega intervala (signal TL je na 1) vrnemo v začetno stanje. Če pa je signal C enak 1 (nočni način), gre vezje v stanje S3 in tako v nočni način delovanja. V tem stanju ostane tako dolgo, dokler je C na 1. Ko gre C na 0, se vrnemo v stanje S2. V vsakem stanju je določena tudi kombinacija izhodnih vrednosti, s katerimi krmilimo luči na semaforjih. Iz diagrama prehajanja stanj dobimo pravilnostno tabelo (tabela 7.3-1) za sekvenčno vezje. V diagramu so prikazane samo tiste vhodne kombinacije, ki povzročijo prehod v drugo stanje.



Slika 7.3-1. Diagram prehajanja stanj.

Tabela 7.3-1. Pravilnostna tabela za sekvenčno vezje.

y_1	y_2	C	TL	TS	y^+_1	y^+_2	HR	HY	HG	PR	PG
0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	1	1	0	1	0	0	1	1	0
0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	1	0	0	0	0	1	1	0
0	0	1	1	0	0	1	0	0	1	1	0
0	0	1	1	1	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0	1	0	1	0
0	1	0	1	0	1	0	0	1	0	1	0
0	1	0	1	1	1	0	0	1	0	1	0
0	1	1	0	0	0	1	0	1	0	1	0
0	1	1	0	1	1	0	0	1	0	1	0
0	1	1	1	0	1	0	0	1	0	1	0
0	1	1	1	1	1	0	0	1	0	1	0
1	0	0	0	0	1	0	1	0	0	0	1
1	0	0	0	1	1	0	1	0	0	0	1
1	0	0	1	0	0	0	1	0	0	0	1
1	0	0	1	1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	1	0	0	0	1
1	0	1	0	1	1	0	1	0	0	0	1
1	0	1	1	0	1	1	1	0	0	0	1
1	0	1	1	1	1	1	1	0	0	0	1
1	1	0	0	0	1	0	0	1	0	0	1
1	1	0	0	1	1	0	0	1	0	0	1
1	1	0	1	0	1	0	0	1	0	0	1
1	1	0	1	1	1	0	0	1	0	0	1
1	1	1	0	0	1	1	0	1	0	0	1
1	1	1	0	1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	0	0	0	0	0
1	1	1	1	1	1	1	0	1	0	0	1

Sekvenčno vezje

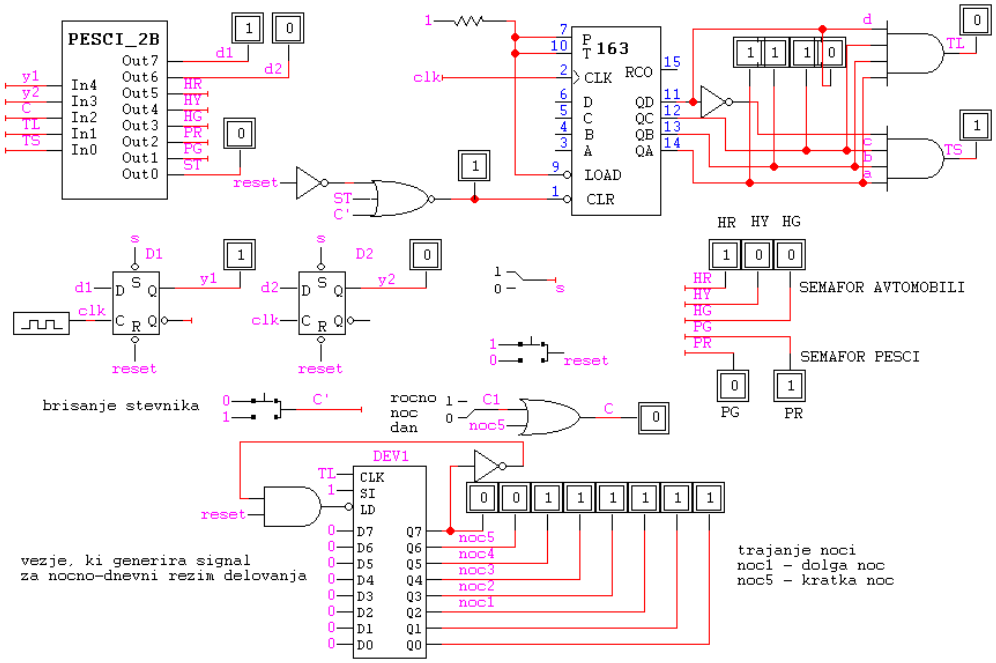
Sekvenčno vezje tipa Moore je srce našega vezja. Ker imamo štiri stanja, smo ga realizirali z dvema pomnilnima celicama D in krmilnim vezjem "PESCI_2B". Ker bi realizacija kombinacijskega vezja s standardnimi elementi zahtevala veliko standardnih vrat, smo ga raje izvedli z vezjem PROM. Postopek za tvorjenje vezja PROM je podan v poglavju 5.7.2. Vezje "PESCI_2B" ima pet vhodnih in osem izhodnih priključkov. Na vhode pripeljemo spremenljivke y_1, y_2, C, TL in TS, na izhodih pa dobimo signale HR, HY in HG, ki krmilijo luči semaforjev za avtomobile, signala PR in PG, ki krmilita luči semaforjev za pešce, ter krmilna signala za pomnilni celici d_1 in d_2. Izhodni signal ST potrebujemo za brisanje časovnika, ki določa dolžino kratkega ter dolgega časovnega intervala. Signal ST se postavi na ena pri vsakem prehodu iz enega v drugo stanje in tako zagotovi postavljanje časovnika v začetno stanje.

Zahtevo po utripanju rumene (semafor za avtomobile) in zelene luči (semafor za pešce) lahko realiziramo na več načinov. Zaradi poenostavitve vezja smo utripanje luči izvedli tako, da smo znotraj stanja S3 vpeljali različne kombinacije izhodnih signalov. To smo prikazali samo v tabeli prehajanja stanj (tabela 7.3-1), ne pa v diagramu na sliki 7.3-1. Ko smo v stanju S3, gorita rumena za avtomobile in zelena za pešce. Obe luči za trenutek ugasneta, ko postane eden od signalov za kratki ali dolgi časovni interval enak ena, nato se luči ponovno prižgeta. Na ta način smo utripanje v nočnem načinu delovanja izvedli zelo enostavno, v realnem vezju pa bi bilo tako utripanje prepočasno.

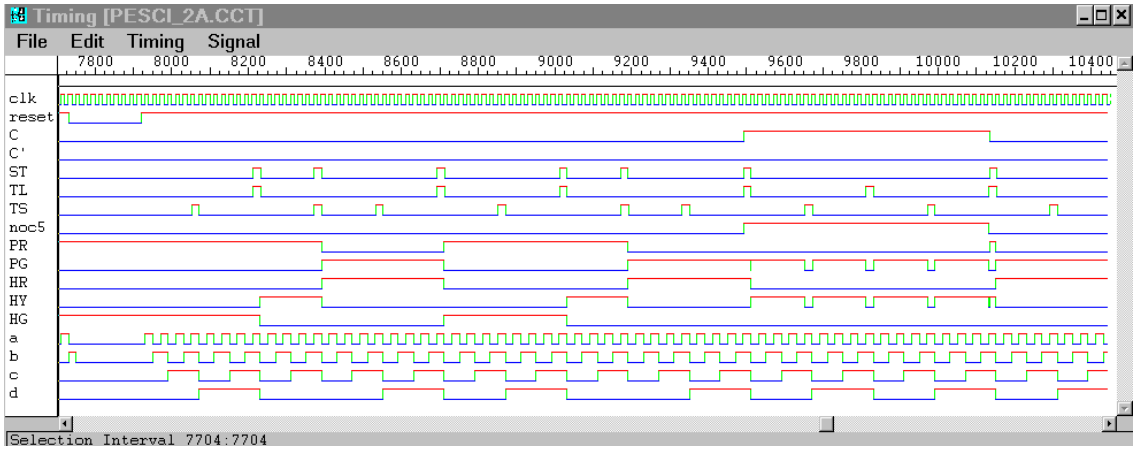
Vezje

Celotno vezje je prikazano na sliki 7.3-2 , rezultat simulacije pa na sliki 7.3-3.

Načrt vezja:



Slika 7.3-2. Vezje za semaforiziran prehod za pešce 2.



Slika 7.3-3. Simulacijski prikaz krmilnih signalov.

7.4 Projekt 4: Semaforizirano križišče

Logic Simulation of Traffic Controller State Machine

1. Laboratory Objectives

In this tutorial, you will gain experience in using LogicWorks to design and simulate a complete hardware subsystem, in this case, the Highway/Farmroad traffic light controller. In particular, you will:

- Construct the subsystems of the traffic light controller in pieces, fully simulating each piece before advancing to the next piece and the integration of pieces.
- Use a LogicWorks PLA for implementing the Next State and Output functions of a FSM.
- Develop debugging and troubleshooting skills.
- Place additional logic and probes as necessary to simplify FSM debugging.
- Appreciate the importance of being able to simulate and debug a circuit before realizing it as actual hardware.

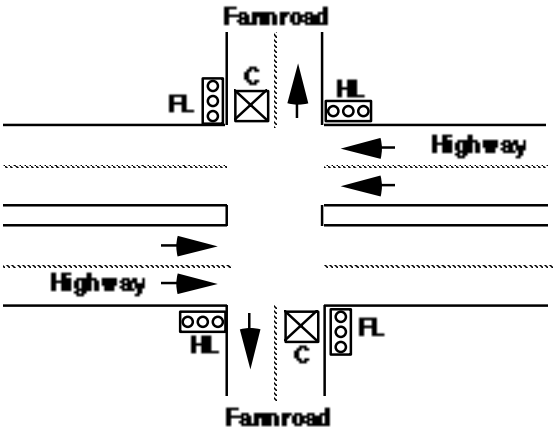


Figure 1. Highway/Farmroad traffic light situation

2. Problem Specification

"A busy highway is intersected by a little used farm road, as shown in Figure 1. Detectors are placed along the farm road to a raise the signal C as long as a vehicle is waiting to cross the highway. The traffic light controller should operate as follows. As long as no vehicle is detected on the farm road, the lights should remain green in the highway direction. If a vehicle is detected on the farm road, the highway lights should transition from yellow to red, allowing the farmroad lights to become green. The farmroad lights stay green only as long as a vehicle is detected on the farmroad, and never longer than a set interval to allow the traffic to flow along the highway. If these conditions are met, the farmroad lights transition from green to yellow to red, allowing the highway lights to return to green. Even if vehicles are waiting to cross the highway, the highway should remain green for a set interval. You may assume that there is an external timer that once set via the control signal ST (set timer), will assert the signal TS after a short time interval has expired (used for timing yellow lights) and TL after a long time interval (for green lights). The timer is automatically reset when ST is asserted."

The inputs and outputs of the finite state machine can be summarized as follows:

Input Signal	**Description**
reset	place controller in initial state
C	detects vehicle on farmroad in either direction
TS	short timer interval has expired
TL	long timer interval has expired
Output Signal	**Description**
HG, HY, HR	assert green, yellow, red highway lights
FG, FY, FR	assert green, yellow, red farmroad lights
ST	commence timing a long or short interval

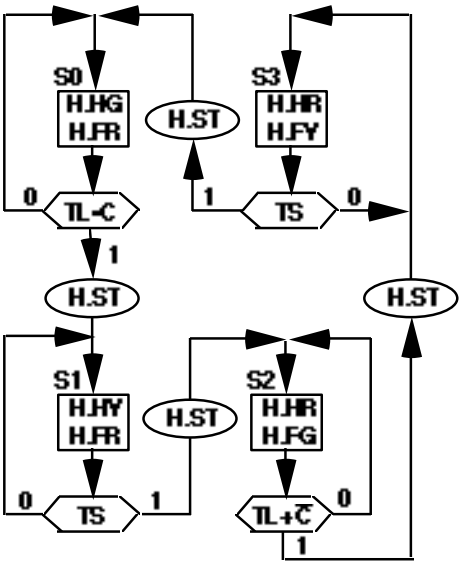
It is fairly easy to convince yourself that there are four unique configurations of the controller:

State	**Description**
S0	Highway green (farmroad red)
S1	Highway yellow (farmroad red)
S2	Farmroad green (highway red)
S3	Farmroad yellow (highway red)

The basic sequencing is S0, S1, S2, S3, and repeat. A reset signal places the controller in state S0, with the highway green and the farmroad red. Reset should also begin the timer. From the problem specification, the controller should stay in state S0 as long as no vehicle is waiting on the farmroad. Even if a vehicle is waiting, the highway is guaranteed to stay green for the long time interval. Thus, the conditions for advancing from S0 to S1 are TL is asserted and C is asserted. In all other cases, the controller should remain in S0.

The controller should remain in S1 for the short time interval before advancing to S2. The behavior of S3, with its transition to S0, is very similar. For S2, the farmroad green state, the exit conditions are: (1) a long time interval has expired, whether or not any cars are still waiting, or (2) there are no more vehicles waiting to cross the intersection. This can be expressed as the condition TL + C. Under any other circumstances, we remain in state S2.

Figure 2 contains the completed ASM chart. An equivalent state diagram is shown in Figure 3 (the traffic light outputs are not shown).



).

Figure 2. ASM chart

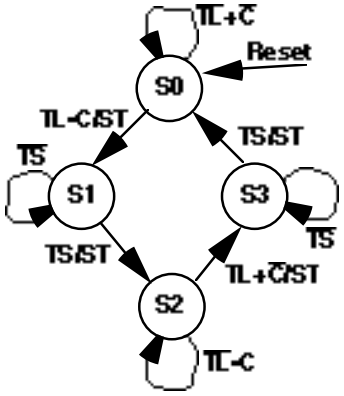


Figure 3. ASM state diagram

3. Problem Decomposition

One possible decomposition of the traffic light controller into more primitive subsystems (many alternatives exist) is the following:

1. controller finite state machine
   1.1. next state/output combinational functions
   1.2. state register
2. short time/long time interval counter
3. car sensor
4. output decoders and traffic lights

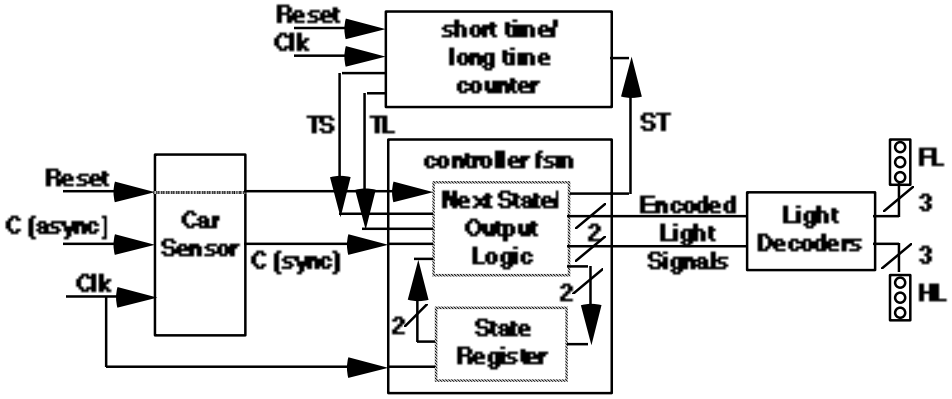


Figure 4. Block diagram.

A block diagram description for this decomposition is shown in Figure 4. The controller FSM takes as input the Reset, Clk, TL, and TS signals, as well as a *synchronized* C signal, and generates the ST signal and encoded light signals (00 = Green, 01 = Yellow, 10 = Red). The interval counter subsystem takes Clk, Reset, and ST as inputs, and generates TL and TS as outputs. The car sensor subsystem has an *asynchronous* sensor input C that it outputs as a synchronized signal. The light decoders take the encoded light control signals and decode them to drive the traffic lights.

It is perfectly reasonable to generate outputs that directly control the lights, rather than the encoded scheme we have chosen. The tradeoff is between reducing the number of outputs and thus wires that need to be routed, and the extra hardware for the decode.

In the way we have drawn Figure 4, it should be obvious that the finite state machine is a Mealy Machine. Is it synchronous or asynchronous? Because the inputs C (sync), TS, and TL are synchronous to the clock, it is a synchronous machine. To be thorough, Reset should also be synchronized.

4. Finite State Machine Description

Once the ASM chart or state diagram has been determined, the next step is to tabulate the state transition table, first using symbolic names for the states and then replacing them with encoded binary patterns. To simplify the ensuing discussion, we will assume the encoding: S0 = 00, S1 = 01, S2 = 11, S3 = 10. It is perfectly valid for you to choose another encoding. The FSM has six inputs: Reset, C, TL, TS, P, Q (the latter are the names we have chosen for the current state signals), and seven outputs: R, S (the names of the next state outputs), ST, HL[1:0], FL[1:0] (recall that the light signals are encoded in two bits—you may choose to directly encode each of these in three bits).

To write down the complete transition table is quite a task: with its six inputs it must have 64 entries! However, we can use don't cares to significantly reduce the number of entries. For example, whenever Reset is asserted, independent of the other inputs, the FSM should return to S0. Through judicious use of don't cares, 32 entries can be reduced to 1! Figure 5 contains the beginnings of an encoded state transition table, containing the transitions from S0. Complete the table for our finite state machine.

5. Implementing a Finite State Machine with a PLA

Once the encoded state transition table is available, the next state is to choose an implementation strategy for the next state (R, S) and output (ST, HL1, HL0, FL1, FL0) combinational logic, as a function of the inputs (Reset, C, TL, TS) and the current state (P, Q). For the purposes of this *LogicWorks* laboratory, we will use a PLA-based design approach.

Specifying a PLA is not unlike describing a PROM as we did in Laboratory #1. Figure 6 shows the new PLA dialog, which is found under the **Options** menu when **New PROM or PLA …** selected. Select PLA Active High (this is the default), provide a name, number of inputs, and number of outputs, and check "Allow reload". An active high PLA is simply one whose outputs are asserted high when the inputs match the associated product terms. It is very important to check "Allow reload": this will enable you to make changes to the PLA specification as you uncover errors during the verification of your design. Select the PLA within your schematic, click on **Edit PROM or PLA …** under the **Options** menu, and you will be presented the PLA dialogs once again.

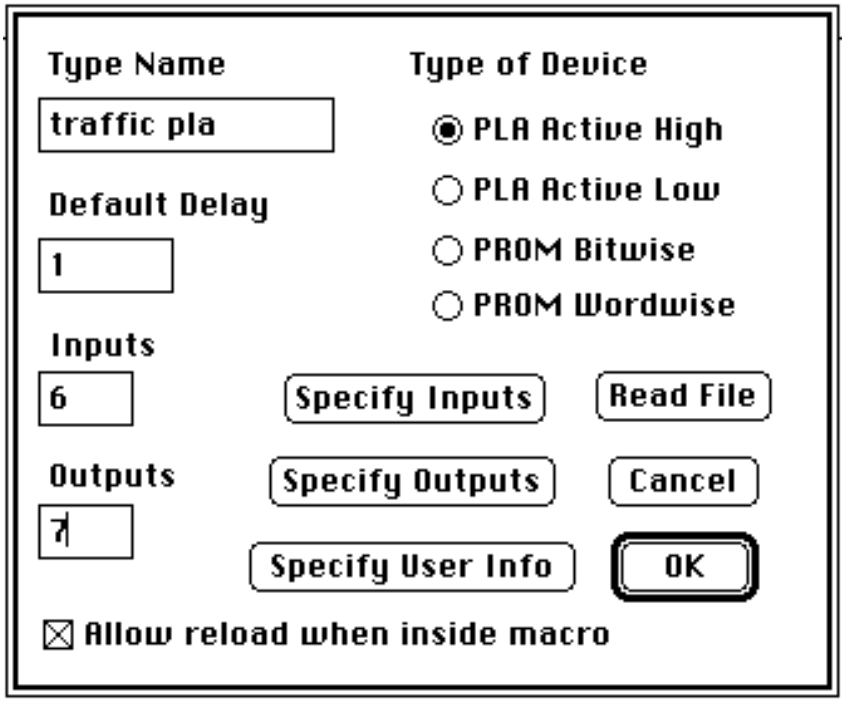


Figure 5. PLA dialog.

You are now ready to specify the names of the inputs. The dialog is shown in Figure 7. Simply enter the names of the input signals, starting with input #0 and working up to input #5. Advance to the next input to be named by clicking on the Next button. The numbering of the inputs is important, and will be used in specifying the products terms that define the outputs. When you are done with the inputs, click on OK to return you to the main PLA dialog. We are now ready to specify the outputs.

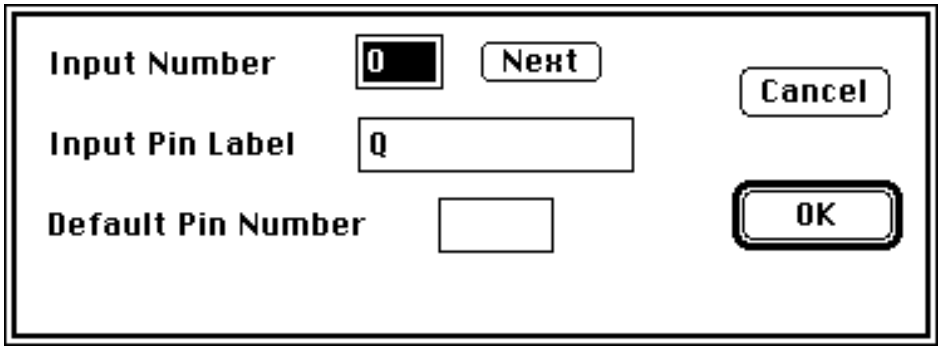


Figure 6. I/O table.

Specifying outputs is much like specifying inputs. You should name each output, and you advance to the next one by clicking on the Next button. In addition, you specify the product terms that cause the output to be asserted. Think of these as patterns against which the input values are compared. If any pattern matches, then the output is asserted either high or low, depending on your original specification of the type of the PLA device. Figure 8 shows the dialog for the FL1 output. Click the cursor in the scrolling box at the bottom of the dialog, and it will turn into an insertion cursor, allowing you to enter one product term per line (after entering the 0, 1, or X for each of the six inputs, the cursor will automatically advance to the next line). The first product term corresponds to the input combination RESET = 1, C = X, TL = X, TS = X, P = X, Q = X. Note that the left to right numbering of inputs goes from highest (RESET) to lowest (Q) input number.

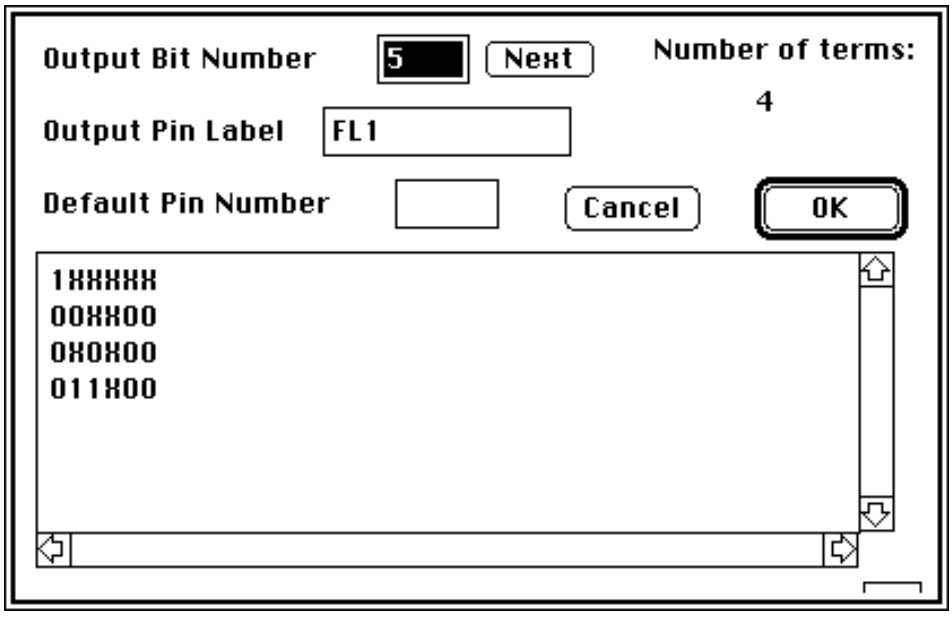


Figure 7. The dialog for the FL1 output.

You will be prompted to place your new PLA in one of your open libraries. The PLA will be available in the selected library under the **Libraries** menu.

6. Constructing the Traffic Light Controller in an Hierarchical Fashion

While the traffic light controller is not a particularly complex system, it still makes sense to build it up piece by piece, thoroughly testing each piece before advancing to the next. It also makes sense to make liberal use of LogicWork's probes and switches to verify proper operation of the component.

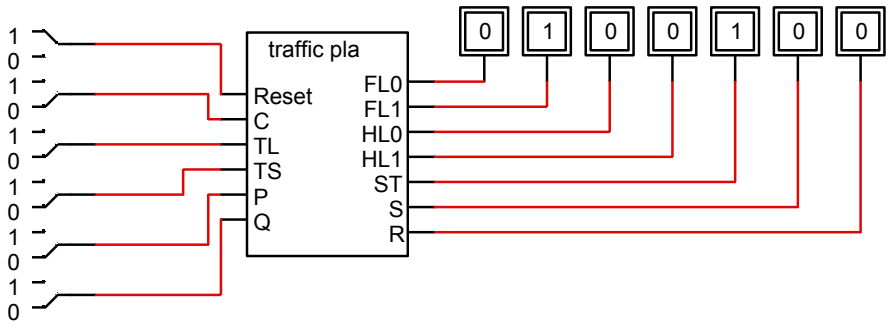


Figure 9. PLA

For example, Figure 9 shows a completely “instrumented” traffic PLA. It should be possible to verify proper operation of the combinational logic before adding the state register and the additional complexities of sequential logic. Once the next state and output functions have been verified, you can advance to adding the state registers, as shown in Figure 10. The state register is implemented by dual D-flip-flops in a 7474 TTL package. With the clock free running, you should be able to verify proper operation of the finite state machine for a variety of different input combinations.

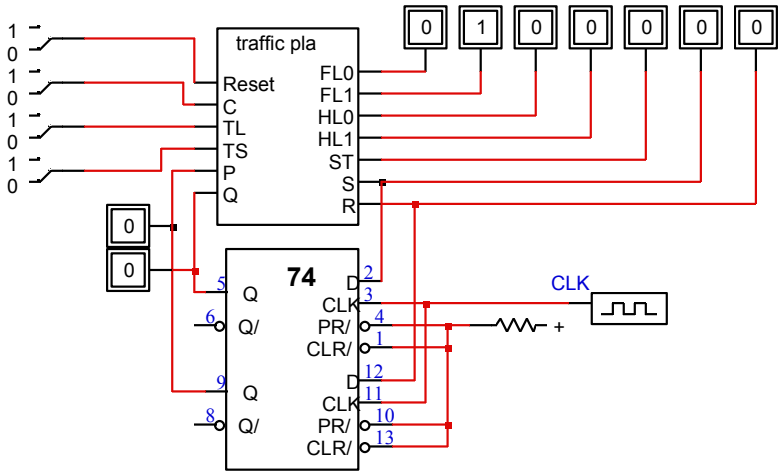


Figure 10. State register

7. External Inputs and Outputs

By this point, the core finite state machine should have been thoroughly tested. You are now ready to tackle the car sensor input and the traffic light decoders.

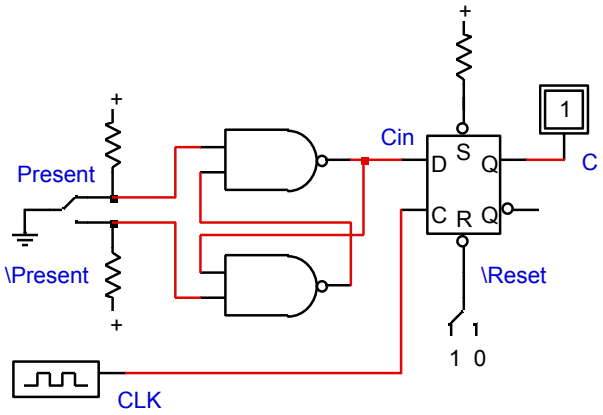


Figure 11. Trafic lights decoder Implementation

You have seen a way to build the car sensor already: the GO switch of the previous tutorial. This circuit combines the elements of a debounced switch with a synchronizer. Figure 11 shows a potential implementation. The switch is currently in the “car present” position. Sketch or print out the simulation waveform that verifies that the sensor input is being debounced and synchronized in Figure 12.

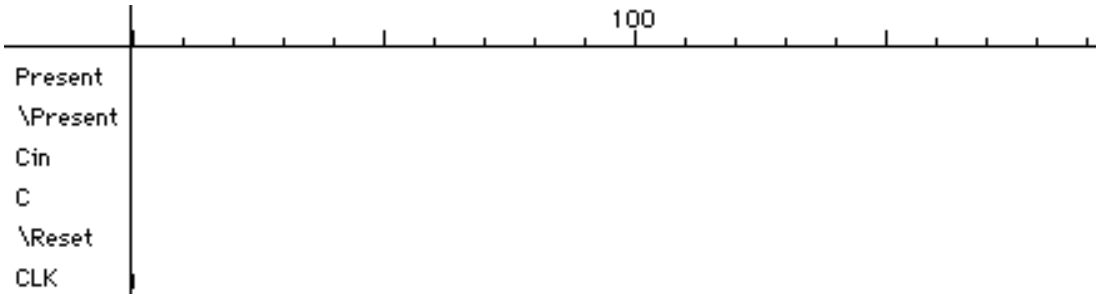


Figure 12. Waveform of the simulation

The light decoders are straightforward. Use 2-to-4 decoders, such as the 74139, which conveniently contains two.

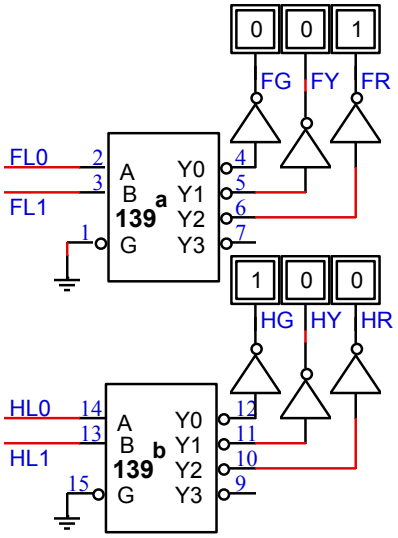


Figure 13 contains the necessary logic.

8. Interval Timer

The last remaining component is the interval timer, designed to generate the signals TL and TS after being set by ST. There are many different ways to implement this, perhaps the simplest being the use of a counter and external decode logic. The counter is cleared when ST is asserted, and the TL and TS are asserted by the external logic when the counter counts up to the appropriate threshold value. For the purposes of this discussion, we will assume that the TS is asserted when the 4-bit counter reaches 0111 and TL is asserted when it reaches 1111.

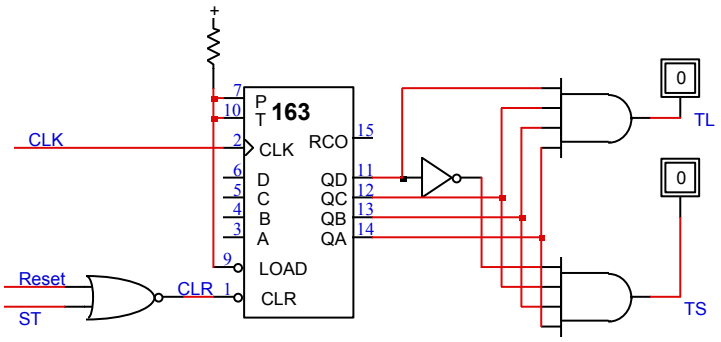


Figure 14. Synchronous up-counter

Figure 14 shows the logic using a 74163 synchronous up-counter. Note the combination of the ST signal and the RESET signal at the CLR input to the counter. This is not strictly necessary, because regardless of the state the counter comes up in when powered on, it will eventually cycle through the states that cause TS and TL to be asserted.

9. Putting It All Together

You are now ready to integrate the pieces into a single design. For more complex circuits, it makes sense to construct each component as its own LogicWorks macro. But as long as the whole design fits within a single schematic page, this is no strictly necessary. The traffic light controller is simple enough to fit on one page.

The following is a suggested integration plan. Other sequences may be just as effective. Start with the PLA, then construct the finite state machine by adding the state registers and feedback. Adding the traffic light decoders is probably the most reasonable next step; it certainly eases the task of verifying that the lights are cycling correctly. Next, wire up the interval counter. This allows you test proper functionality by single or slow stepping the clock, controlling the car sensor input with a simple switch. The final step integrates the car sensor circuitry. Now you can run the simulation at full speed, setting the debounced car sensor switch to simulate the behavior of waiting cars at the farmroad. (NOTE: You may want to slow down the clock used in the simulation. Simply select the clock icon, and click on **Set Params …** under the **Options** menu. Increase the high and low times of the clock from the default value of 10 simulation time units.)

10. Review

In this laboratory, we have used LogicWorks to simulate the operation of a traffic light controller. We have built a simple datapath consisting of the timer interval counter, the debounced car input sensor, and the traffic light output decoders. In addition, we learned how to “program” a PLA for the combinational logic functions of the next state and the outputs, and to combine this PLA with a storage register to form a core finite state machine.

How easy was it for you to validate the correctness of your design through simulation? Did you have any difficulties using LogicWorks, such as its handling of start-up states or don’t cares? What aspects of real hardware, especially timing, are not modeled by LogicWorks?

So now you can actually go ahead and build the traffic light controller in hardware. While building and debugging this circuit, consider the following questions:

• what is the relative easy of debugging and troubleshooting in the hardware versus software environments?

• how close does simulated behavior come to the actual behavior you observe in the hardware?

• how useful was it to be able to design and simulate the logic before building it?

7.5 Projekt 5: Vezje za krmiljenje dviganja/spuščanja avtomatske zapornice 1

Realizirali smo vezje, ki krmili dviganje/spuščanje avtomatske zapornice. Zapornico upravljamo z daljinskim upravljalnikom.

V območju zapornice je nameščen senzor Senzor (tabela 7.5-1), ki ugotavlja, ali je v območju zapornice prepreka (v času, ko se zapornica dviga ali spušča). V primeru, ko ovira v območju zapornice prekine snop senzorja, se mora dviganje ali spuščanje zapornice takoj prekiniti. Po odstranitvi ovire se dviganje ali spuščanje nadaljuje. Zapornica ima nameščeni še položajni stikali Z in O; ti signalizirata vezju, kdaj zapornica doseže en ali drug končni položaj.

Tabela 7.5-1. Opis vhodnih stikal zapornice.

Senzor (stikalo)	**Funkcija senzorja (stikala)**
Senzor	ugotavlja prepreko v območju zapornice
O	položajno stikalo, ki določa končni položaj odprte zapornice
Z	položajno stikalo, ki določa končni položaj zaprte zapornice

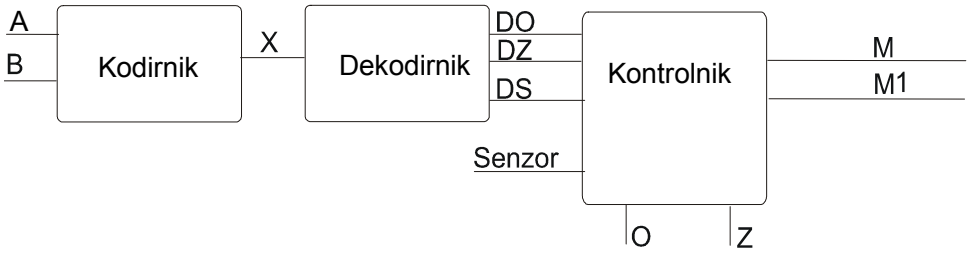
Vezje upravljamo s stikaloma A in B, nameščenima v namišljenem daljinskem upravljalniku. Kombinacije signalov A in B določajo, ali želimo dviganje, spuščanje ali njeno prekinitev (tabela 7.5-2). Senzor je na 1, ko je v območju zapornice prepreka. Položajni stikali O in Z sta na 1, ko je zapornica v enem ali v drugem končnem položaju. Če je katerikoli od vhodov Senzor, O ali Z na 1, mora kombinacija na izhodu vezja zahtevati prekinitev odpiranja ali zapiranja zapornice. Vezje ima dva izhoda, izhod M in izhod M1. Kombinacija M M1 nam določa smer vrtenja motorja, tj. ali naj se zapornica dviga ali spušča.

Tabela 7.5-2. Tabela vrednosti posameznih vhodnih signalov.

<table>
<tr><th>Primarni vhodi</th><th>A B</th><th colspan="2">Sekundarni vhodi</th><th>Senzor</th><th>Stikalo O</th><th>Stikalo Z</th></tr>
<tr><td rowspan="2">Signal za odpiranje zapornice</td><td rowspan="2">0 1</td><td rowspan="2">Prepreka</td><td>DA</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>NE</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td rowspan="2">Signal za zapiranje zapornice</td><td rowspan="2">1 0</td><td rowspan="2">Skrajni položaj</td><td>DA</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>NE</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Signal za prekinitev</td><td>1 1</td><td colspan="5" rowspan="6">Primarni vhodi so realizirani z daljinskim krmilnikom.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Izhodi vezja</td><td>M M1</td></tr>
<tr><td>Dviganje zapornice</td><td>0 1</td></tr>
<tr><td>Spuščanje zapornice</td><td>1 0</td></tr>
<tr><td>Prekinitev</td><td>1 1</td></tr>
</table>

Blokovna shema

Vezje sestavljajo tri komponente (slika 7.5-1): *Kodirnik, Dekodirnik in Kontrolnik*. Kodirnik generira željeno sekvenco in jo odda Dekodirniku. Ta jo razpozna in določi vrednosti na izhodih DO, DZ in DS. Vrednosti preda Kontrolniku, ki preverja tudi stanja signalov senzorja Senzor in položajnih stikal O in Z . Kontrolnik omenjene signale obdela in z izhodoma M in M1 krmili motor. Vsi signali v vezju so opisani v tabeli 7.5-3.



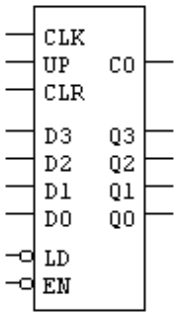
Slika 7.5-1. Blokovni diagram vezja za krmiljenje zapornic.

Tabela 7.5-3. Opis signalov v vezju.

A	vhodni signal	O	položajno stikalo-zapornica je odprta
B	vhodni signal	Z	položajno stikalo- zapornica je zaprta
X	sekvenca	Senzor	senzor za prepreko
DO	izhod Dekodirnika – dviguj	M	izhod na motor
DZ	izhod Dekodirnika – spuščaj	M1	izhod na motor
DS	izhod Dekodirnika – stoj		

Kodirnik

Je vezje, ki simulira delovanje daljinskega upravljalnika (slika 7.5-3). Sestavljajo ga štiribitni števnik in štirje multipleksorji. Tri smo uporabili za generiranje sekvence, enega pa za izbiranje sekvence (sekvenco izberemo s kombinacijo stikal A in B). Števnik (slika 7.5-2) ima štiri vhode, D0, D1, D2, in D3, ter štiri izhode, Q0, Q1, Q2, Q3, stikalo za vklop EN, stikalo za odštevanje UP, priključek za urin signal CLK, stikalo za brisanje CLR in stikalo za ustavitev štetja LD. CO je izhodni priključek števnika, ki se na 1 postavi vedno, ko števnik preide v začetno stanje 0000. Multipleksorji imajo dva naslovna in štiri podatkovne priključke.



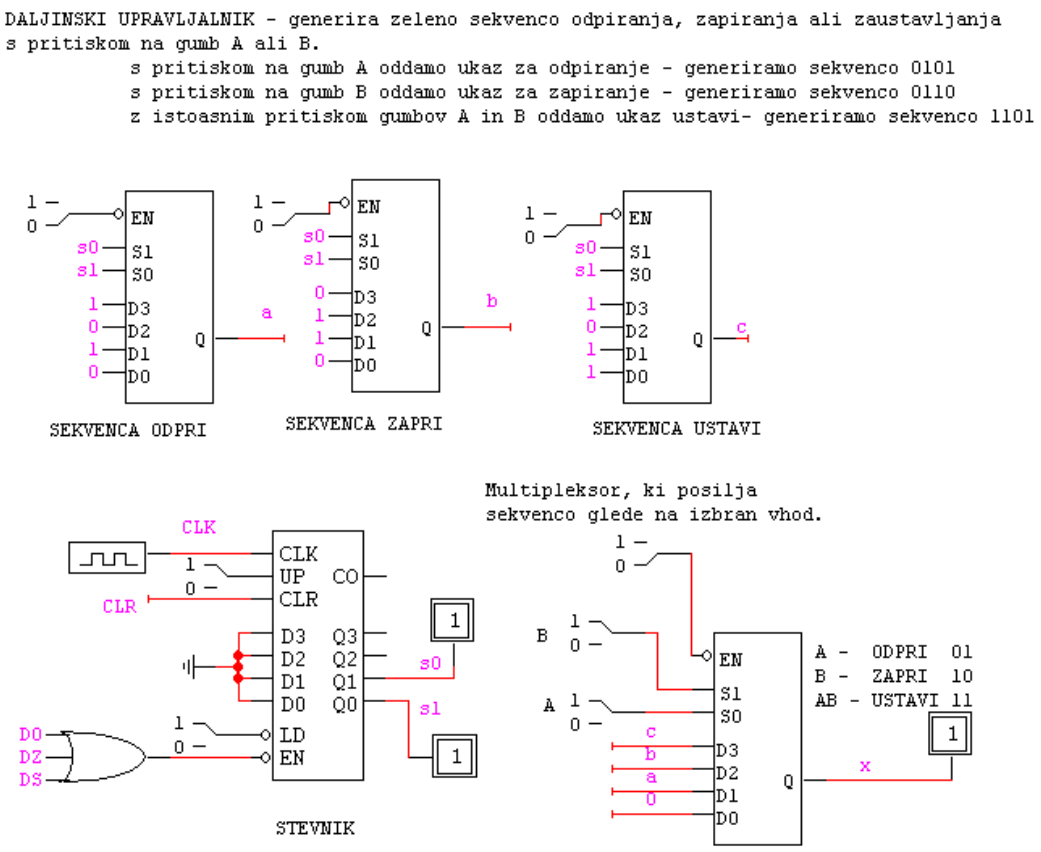
Slika 7.5-2. Števnik gor/dol.

Ker za štetje uporabljamo štiribitno kodo, zadostujeta dva bita. Na vezju zato uporabimo le dva izhoda, Q0 in Q1 Vhodne priključke D0, D1, D2, in D3 povežemo na 0. Priključek CLR povežemo s stikalom Clear, ki ga povežemo tudi na Dekodirnik, tako da z brisanjem Dekodirnik postavimo v začetno stanje, hkrati pa tudi začetek generiranja sekvence z mulitpleksorji. Priključka LD ne potrebujemo in je ves čas na 1. Priključek EN se postavi na 1, kadar katero izmed vezij razpozna pravilno sekvenco in tako zaustavi nadaljnje pošiljanje sekvenc.

Na naslovne priključke povežemo izhode števnika, na podatkovnih (D0, D1, D2, in D3) pa nastavimo vrednosti (0 ali 1); tako določimo posamezno sekvenco. Ko števnik prične šteti, na izhod z multipleksorjem zaporedoma preklapljamo vrednosti, ki jih predhodno nastavimo na naslovnih priključkih. Na ta način na izhodu generiramo vnaprej nastavljene sekvence. S kombinacijo stikal A in B (tabela 7.5-4), ki sta priključeni na naslovne vhode multipleksorja, določimo, katero od sekvenc a, b in c bomo prenesli na izhod oz. katero od sekvenc bo navidezni daljinski upravljalnik oddajal.

Tabela 7.5-4. Generiranje vhodnih sekvenc glede na kombinacijo stikal A in B.

KOMBINACIJA STIKAL A IN B	GENERIRANA SEKVENCA
01 – pritisnjeno je stikalo A	0101
10 – pritisnjeno je stikalo B	0110
11 – pritisnjeni sta obe stikali	1101



Slika 7.5-3. Načrt Kodirnika.

Dekodirnik

Dekodirnik (slika 7.5-4) smo realizirali s tremi sekvenčnimi vezji, ki razpoznajo eno od treh sekvenc (a, b ali c). Ko Dekodirnik razpozna katero od sekvenc, se pošiljanje zaustavi s stikalom CLEAR in zaustavi se tudi števnik. Na izhodih Dekodirnika dobimo kombinacijo vhodov za Kontrolnik. Na Kodirniku lahko spustimo stikalo in ustrezna kombinacija na izhodih Dekodirnika se ohrani vse do ponovnega pritiska na stikali A in B. Ko izberemo želeno kombinacijo stikal A in B, sprožimo pošiljanje naslednje sekvence. Ko je sekvenca v celoti odposlana, jo razpozna eden izmed avtomatov Dekodirnika in ustrezni izhod (DO, DZ ali DS) se postavi na 1. Izhodi se lahko postavijo na 1 le, če števnik ni v začetnem stanju. Z brisanjem števnika izhode dekodirnika postavimo na 0. Sekvenčna vezja v Dekodirniku za razpoznavanje vhodnih sekvenc smo realizirali s pomnilnimi celicami JK. Delovanje sekvenčnih vezij je predstavljeno s tabelami prehajanja stanj ter s pomnilnimi in izhodnimi enačbami, ki sledijo.

a) Sekvenčno vezje za razpoznavanje sekvence 0101 (sekvenca za odpiranje). Izhod se postavi na 1, ko vezje sekvenco razpozna, drugače je izhod na 0.

Tabela prehajanja stanj:

Tabela prehajanja stanj			Kodirana tabela prehajanja stanj		
Sedanje stanje	Naslednje stanje / izhod		Sedanje stanje	Naslednje stanje / izhod	
	0	1		0	1
A	B/0	A/0	0 0	01/0	00/0
B	A/0	C/0	0 1	00/0	10/0
C	D/0	A/0	1 0	11/0	00/0
D	A/0	A/1	1 1	00/0	00/1

Primarni vhod	Sekundarna vhoda	Sekundarna izhoda	Celica 1	Celica 2	Izhod z
x	y_1 y_2	y_1^+ y_2^+	J_1 K_1	J_2 K_2	Z
0	0 0	0 1	0 X	1 X	0
0	0 1	0 0	0 X	X 1	0
0	1 0	1 1	X 0	1 X	0
0	1 1	0 0	X 1	X 1	0
1	0 0	0 0	0 X	0 X	0
1	0 1	1 0	1 X	X 1	0
1	1 0	0 0	X 1	0 X	0
1	1 1	0 0	X 1	X 1	1

Pomnilne enačbe (minimizacija pomnilnih funkcij):

J_1:

y_1,y_2 \ x	0	1
00	0	0
01	0	1
11	X	X
10	X	X

$J_1 = xy_2$

K_1:

y_1,y_2 \ x	0	1
00	X	X
01	X	X
11	1	1
10	0	1

$K_1 = x + y_2$

J_2:

y_1,y_2 \ x	0	1
00	1	0
01	X	X
11	X	X
10	1	0

$J_2 = /x$

K_2:

y_1,y_2 \ x	0	1
00	X	X
01	1	1
11	1	1
10	X	X

$K_2 = 1$

Enačba izhoda:

$Z = xy_1y_2$

b) Sekvenčno vezje za razpoznavanje sekvence 0110 (sekvenca za zapiranje). Izhod se postavi na 1, ko vezje sekvenco razpozna, drugače je izhod na 0.

Tabela prehajanja stanj:

Tabela prehajanja stanj			Kodirana tabela prehajanja stanj		
Sedanje stanje	Naslednje stanje / izhod		Sedanje stanje	Naslednje stanje / izhod	
	0	1		0	1
A	B/0	A/0	0 0	01/0	00/0
B	A/0	C/0	0 1	00/0	10/0
C	A/0	D/0	1 0	00/0	11/0
D	A/1	A/0	1 1	00/1	00/0

Primarni vhod	Sekundarna vhoda	Sekundarna izhoda	Celica 1	Celica 2	Izhod z
x	y_1 y_2	y_1^+ y_2^+	J_1 K_1	J_2 K_2	Z
0	0 0	0 1	0 X	1 X	0
0	0 1	0 0	0 X	X 1	0
0	1 0	0 0	X 1	0 X	0
0	1 1	0 0	X 1	X 1	1
1	0 0	0 0	0 X	0 X	0
1	0 1	1 0	1 X	X 1	0
1	1 0	1 1	X 0	1 X	0
1	1 1	0 0	X 1	X 1	0

Pomnilne enačbe:

J_1:

y_1,y_2 \ x	0	1
00	0	0
01	0	1
11	X	X
10	X	X

$J_1 = xy_2$

K_1:

y_1,y_2 \ x	0	1
00	X	X
01	X	X
11	1	1
10	1	0

$K_1 = /x + y_2$

J_2:

y_1,y_2 \ x	0	1
00	1	0
01	X	X
11	X	X
10	0	1

$J_2 = (x + y_1)\,(/x + /y_1)$

K_2:

y_1,y_2 \ x	0	1
00	X	X
01	1	1
11	1	1
10	X	X

$K_2 = 1$

Enačba izhoda:

$Z = /xy_1y_2$

c) Sekvenčno vezje ugotovi sekvenco 1101 (sekvenca stop). Če se pojavi sekvenca 1101 in jo razpozna, da na izhod 1, če ne, je izhod na 0.

Tabela prehajanja stanj:

Tabela prehajanja stanj			Kodirana tabela prehajanja stanj		
Sedanje stanje	Naslednje stanje / vhod		Sedanje stanje	Naslednje stanje / vhod	
	0	1		0	1
A	A/0	B/0	0 0	00/0	01/0
B	A/0	C/0	0 1	00/0	10/0
C	D/0	A/0	1 0	11/0	00/0
D	A/0	A/1	1 1	00/0	00/1

Primarni vhod	Sekundarna vhoda	Sekundarna izhoda	Celica 1	Celica 2	Izhod z
x	y_1 y_2	y_1^+ y_2^+	J_1 K_1	J_2 K_2	Z
0	0 0	0 0	0 X	0 X	0
0	0 1	0 0	0 X	X 1	0
0	1 0	1 1	X 0	1 X	0
0	1 1	0 0	X 1	X 1	0
1	0 0	0 1	0 X	1 X	0
1	0 1	1 0	1 X	X 1	0
1	1 0	0 0	X 1	0 X	0
1	1 1	0 0	X 1	X 1	1

Pomnilne enačbe:

J_1:

y_1,y_2 \ x	0	1
00	0	0
01	0	1
11	X	X
10	X	X

$J_1 = xy_2$

K_1:

y_1,y_2 \ x	0	1
00	X	X
01	X	X
11	1	1
10	0	1

$K_1 = y_2 + x$

J_2:

y_1,y_2 \ x	0	1
00	0	1
01	X	X
11	X	X
10	1	0

$J_2 = /x\, y_1 + x\, /y_1$

K_2:

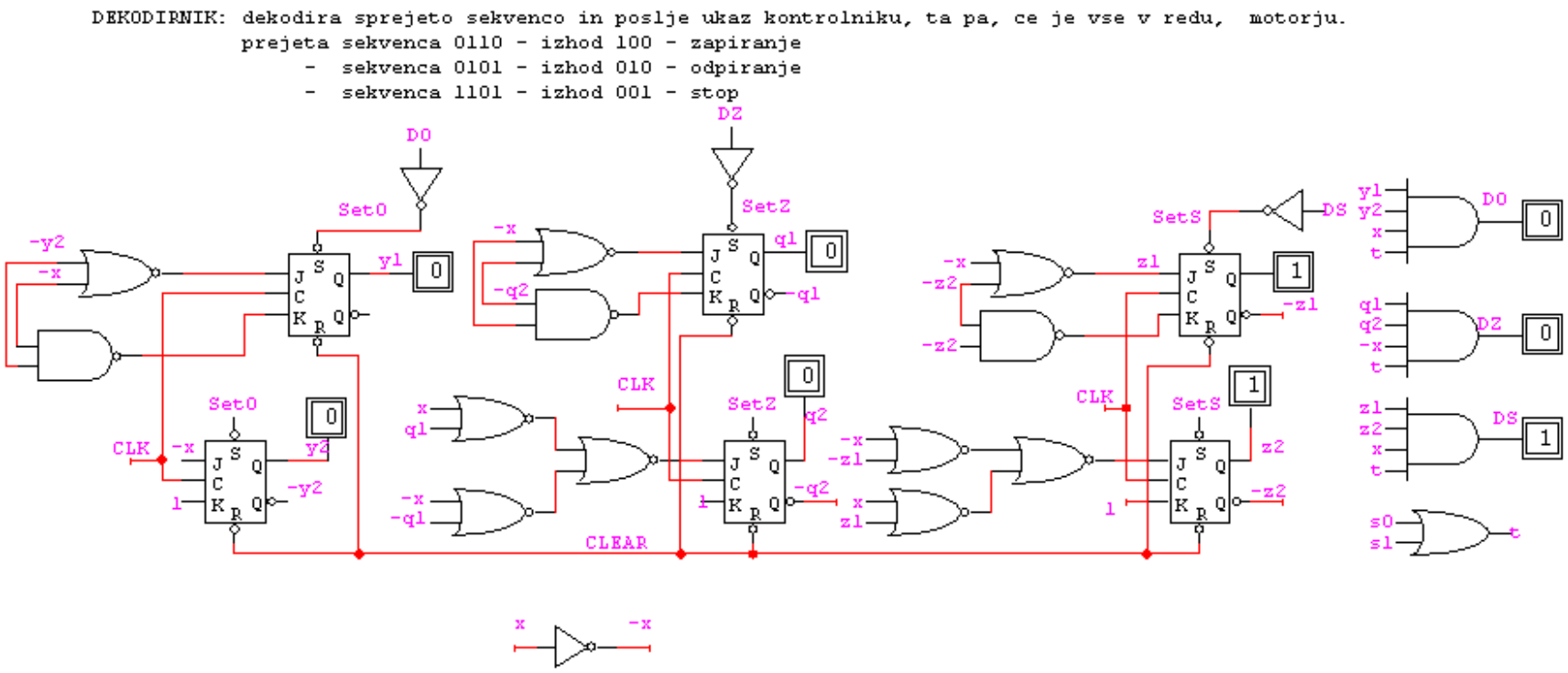
y_1,y_2 \ x	0	1
00	X	X
01	1	1
11	1	1
10	X	X

$K_2 = 1$

Enačba izhoda:

$Z = x\, y_1 y_2$

Vezje Dekodirnika:



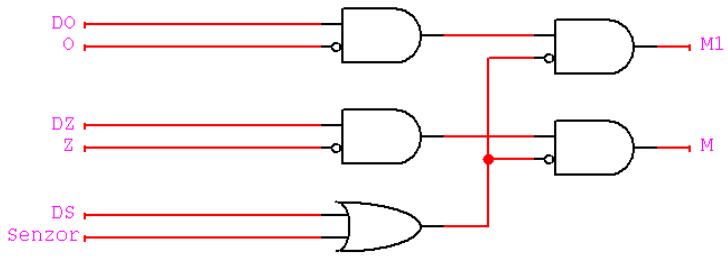
Slika 7.5-4. Načrt Dekodirnika.

Kontrolnik

Kontrolnik (slika 7.5-5) prejema podatke od Dekodirnika in signale s položajnih stikal zapornice O in Z ter senzorja Senzor, ki ugotavlja, ali je v območju zapornice prepreka. Če je katerikoli od zadnjih treh signalov na 1, pomeni, da je v času dviganja/spuščanja zapornice prišlo do prekinitve snopa senzorja ali pa je zapornica že v enem od obeh končnih položajev. V tabeli 7.5-5 so podane kombinacije signalov MM_1, ki krmilijo elektromotor zapornice, da se le-ta dviga, spušča ali stoji. Preprosto kombinacijsko vezje Kontrolnika (slika 7.5-5) bo postavilo izhodno kombinacijo signalov MM1 na 0 0 (stop), če bo kateri od signalov Senzor ali DS na 1. Kombinacija signalov MM1 bo na 0 0 tudi, če bosta hkrati na 1 signala DO in O ali pa DZ in Z. Izhodna kombinacija signalov MM1 bo na 0 1 (dviganje), če bosta signala Senzor in DS oba na 0 ter kombinacija signalov DO in O na 1 0. Izhodna kombinacija signalov MM1 bo na 1 0 (spuščanje), če bosta signala Senzor in DS oba na 0, kombinacija signalov DZ in Z pa na 1 0.

Tabela 7.5-5. Kombinacije signalov MM1, ki krmilijo elektromotor zapornice.

Aktivost zapornice	M M_1
STOP	0 0
DVIGANJE	0 1
SPUŠČANJE	1 0



Slika 7.5-5. Načrt Kontrolnika.

Povezava posameznih komponent v skupno vezje

Vse komponente povežemo v skupno vezje. Pred vsako spremembo na daljinskem upravljalniku Dekodirnik resetiramo s stikalom CLEAR, da se le-ta postavi na začetek sprejemanja sekvence. Istočasno se na 0 postavi tudi števnik.

7.6 Projekt 7: Prikazovanje izbranih telefonskih številk ter avtomatsko pozivanje

Za izdelavo projekta smo uporabili kar tipkovnico iz knjižnice LogicWorks, ki ima 5 izhodov. Na prvih štirih izhodih dobimo vrednosti signalov (tabela 7.6-1), ki nam določajo številko ali znak pritisnjene tipke. Peti izhod je krmilni in se postavi na 1 vedno, ko pritisnemo poljubno tipko na tipkovnici. Krmilni signal je kratek in je na 1 le ob pritisku na katero izmed tipk na tipkovnici. Če tipko držimo pritisnjeno dalj časa, to na trajanje izhodnega impulza nima vpliva (ni večkratnega proženja ob daljšem pritisku na tipko). Realizirali smo vezje, s katerim je mogoče izbrati in prikazati šestmestna števila. Če uporabnik vtipka večje število ali če več kot 5 sekund ne pritisne nobene od tipk na tipkovnici, se zaslon v celoti zbriše.

Tabela 7.6-1. Kombinacije izhodov za posamezne tipke na tipkovnici.

Znak	Koda			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Zaslon

Zaslon smo sestavili s šestimi 7-segmentnimi LED prikazovalniki, ki imajo po štiri vhodne priključke. Če izhodne priključke tipkovnice neposredno povežemo z vhodnimi priključki na 7-segmentnem prikazovalniku, pokaže vrednost nazadnje pritisnjene tipke. Prikazovalnik postavimo na 0, tako da vse vhode priključimo na U_L (0V ali logična 0).

Zadrževalnik

Vrednosti iz enega prikazovalnika v drugega prenesemo s ponovnim pritiskom katere izmed tipk na tipkovnici. Krmilni signal na tipkovnici se ob pritisku na tipko za kratek čas postavi na 1 in z njim lahko prožimo zadrževalnik (slika 7.6-2), ki prenese vrednosti signalov s svojih vhodnih priključkov (D0, D1, D2, D3) na izhodne (Q0, Q1, Q2, Q3). Prenos se izvede vedno, ko se izhod tipkovnice O_5 postavi na 1.

Vsak zadrževalnik je sestavljen iz štirih zadrževalnih celic D. Če je vrednost signala priključka C na 1, sledi izhod Q vhodu D. Ko je vrednost signala priključka C na 0, vrednost izhoda Q ni več odvisna od signala na vhodu D, temveč je enaka vrednosti signala na vhodu D, kot je bila ob trenutku, ko je vrednost signala na C prešla z 1 na 0. V tabeli 7.6-2 je prikazano delovanje zadrževalnika.

Tabela 7.6-2. Tabelarični prikaz delovanja zadrževalnika.

VHODI				KRMILNA PRIKLJUČKA		IZHODI			
D1	D2	D3	D4	C	R'	Q1	Q2	Q3	Q4
X	X	X	X	X	1	0	0	0	0
D1	D2	D3	D4	1	0	D1	D2	D3	D4
D1	D2	D3	D4	0	0	Q1	Q2	Q3	Q4

D1-D2: vhodni priključki.

Q1-Q2: izhodni priključki.

C: omogoči ali onemogoči sledenje izhodnih signalov vhodnim.
R: postavi vrednosti na izhodnih priključkih na 0 neodvisno od vrednosti na vhodnih priključkih.

Vezje

Zadrževalnike smo povezali zaporedno. Med seboj smo povezali vse krmilne priključke C ter vse omogočitvene priključke R. Tako lahko prožimo ali brišemo izhode vseh štirih zadrževalnikov istočasno. Vhode D0, D1, D2, D3 prvega zadrževalnika smo povezali na pripadajoče izhode tipkovnice, krmilne priključke C pa na krmilni izhod tipkovnice O_5. Vse štiri izhode zadrževalnika, Q0, Q1, Q2, Q3, smo povezali na vhode prvega prikazovalnika. Tako smo realizirali del vezja za hranjenje in prikaz prve vtipkane vrednosti na tipkovnici. Preostalih pet prikazovalnikov smo povezali na pripadajoče izhode Q0, Q1, Q2, Q3 preostalih zadrževalnikov, dodatno pa smo njihove izhode povezali tudi na vhodne priključke naslednjega zadrževalnika. S tem smo dosegli, da ob vsakem pritisku poljubne tipke na tipkovnici krmilimo vse zaporedje zadrževalnih celic istočasno, te pa v trenutku proženja sprejmejo vrednosti z izhodov predhodnega zadrževalnika, z izjemo prvega zadrževalnika, ki sprejme vrednosti z izhodov tipkovnice (O_1-O_4). Sprejeti signali se na izhod prenesejo z zakasnitvenim časom d_t, tako da sprejetje vrednosti na vhodih zadrževalnika prehiteva spremembo vrednosti na izhodih predhodnega zadrževalnika.

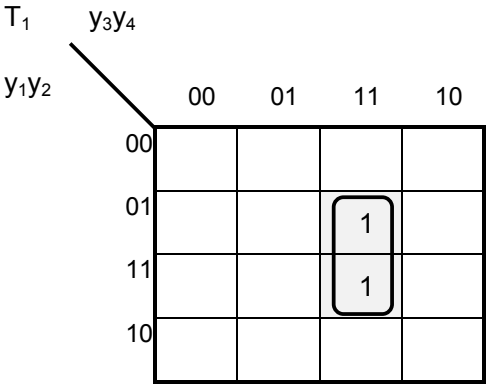
Izhodi zadnjega zadrževalnika v zaporedju so povezani na štirivhodna vrata ALI. Torej, če je vrednost, prenesena iz zadnjega prikazovalnika, različna od 0 0 0 0, dobimo na izhodu vrat ALI logično enico. To pa pomeni, da smo vpisali vrednosti v vse prikazovalnike. Ta signal uporabimo za brisanje vrednosti na prikazovalnikih.

Brisanje pa ni odvisno samo od enega pogoja (tj. vpisane vrednosti v vse prikazovalnike), temveč je odvisno tudi od časa, ki preteče od zadnjega pritiska katere od tipk. Če v času 5 sekund ne pritisnemo nobene od tipk, se vrednosti na prikazovalniku samodejno zbrišejo. Uporabimo števnik, ki nam bo po določenem času na izhodu dal kratkotrajni signal za brisanje prikazovalnikov. Uporabili smo 4-bitni števnik, ki postavi izhod na 1, ko prešteje do vrednosti 1 1 1 1. Realizirani števnik vidimo na sliki 7.6-1. Dolžino trajanja štetja lahko

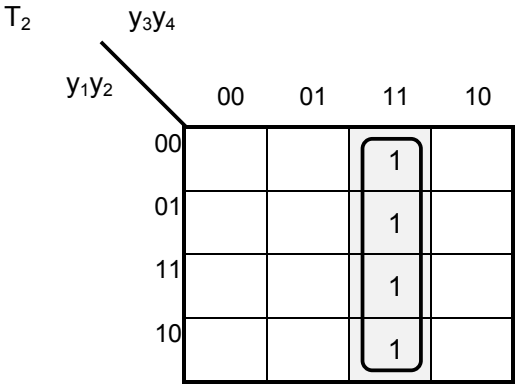
povečamo s serijsko vezavo dveh ali več števnikov. Časovni interval tako lahko prilagodimo zahtevi.

Tabela 7.6-3. Tabela prehajanja stanj 4-bitnega števnika s pomnilnimi celicami T.

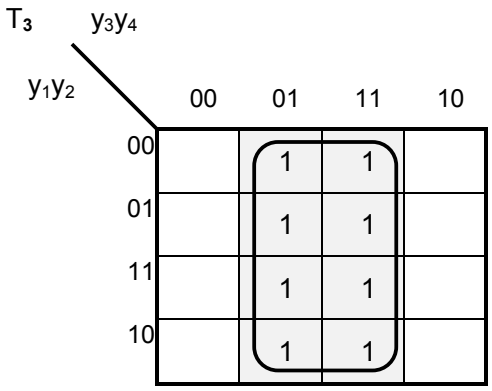
y_1	y_2	y_3	y_4	y_1^+	y_2^+	y_3^+	y_4^+	T_1	T_2	T_3	T_4	z
0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	1	0	0	0	1	1	0
0	0	1	0	0	0	1	1	0	0	0	1	0
0	0	1	1	0	1	0	0	0	1	1	1	0
0	1	0	0	0	1	0	1	0	0	0	1	0
0	1	0	1	0	1	1	0	0	0	1	1	0
0	1	1	0	0	1	1	1	0	0	0	1	0
0	1	1	1	1	0	0	0	1	1	1	1	0
1	0	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	1	0	1	0	0	0	1	1	0
1	0	1	0	1	0	1	1	0	0	0	1	0
1	0	1	1	1	1	0	0	0	1	1	1	0
1	1	0	0	1	1	0	1	0	0	0	1	0
1	1	0	1	1	1	1	0	0	0	1	1	0
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	0	1	1	1	1	1



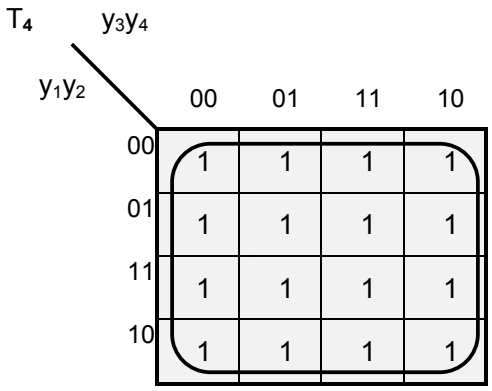
$T_1 = y_2y_3y_4$



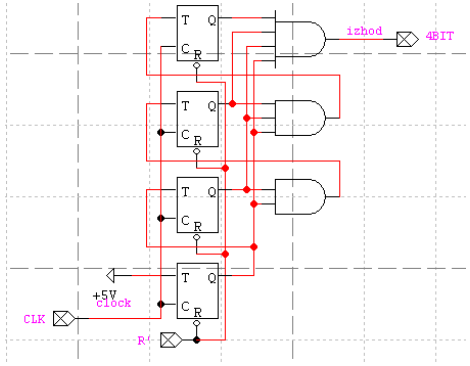
$T_2 = y_3y_4$


T3
y3y4
y1y2
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1

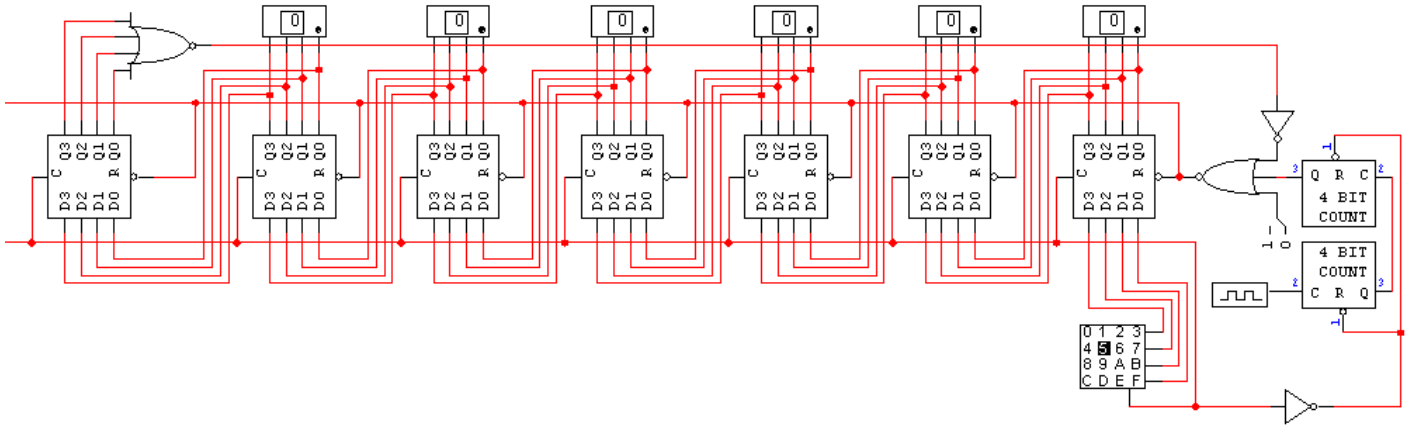
$T_3 = y_4$


T4
y3y4
y1y2
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1

$T_4 = 1$



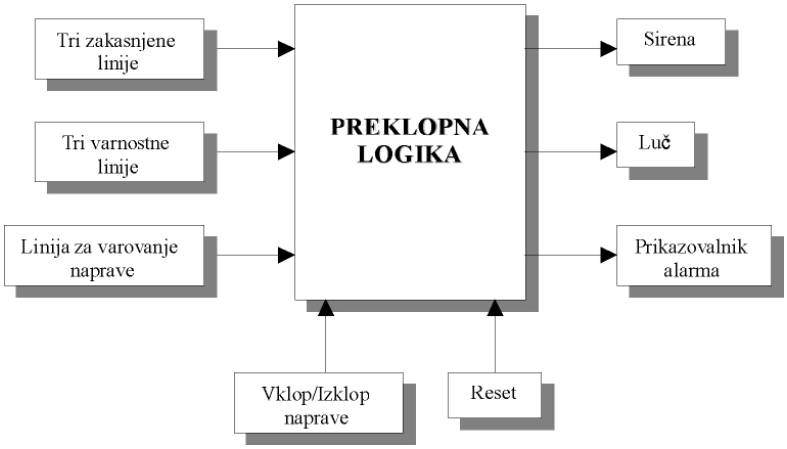
Slika 7.6-1. Realizacija štiribitnega števnika.



Slika 7.6-2. Vezje za prikazovanje in izbiranje telefonskih številk.

7.7 Projekt 8: Preprosta varnostna naprava

Projekt varnostne naprave lahko razdelimo na posamezne bloke (slika 7.7-1),ki skupaj tvorijo varnostno napravo.



Slika 7.7-1. Blokovna shema preproste varnostne naprave.

Preklopna logika:

- pomnilno vezje,
- časovnik za zvočno signalizacijo,
- časovnik za zakasnitve za vstopno/izstopno linijo,
- generator kratkega časovnega impulza,
- generator urinega signala.

Tipala in varnostne linije:

- tri zakasnjena tipala,
- štiri varnostne linije, od katerih je ena za varovanje same alarmne naprave.

Signalizacija:

- sirena,
- luč,
- pomnilnik, ki ohranja stanje alarmne naprave.

Vklop/izklop, reset:

- stikalo za vklop ter izklop naprave,
- stikalo za reset naprave.

Opis vezja za preklopno logiko

Pomnilne celice JK skrbijo za ohranjanje stanj ob sprožitvi tipal, varnostne povezave, krmiljenje luči ter prikazovalnika stanja alarma (slika 7.7-2). Časovnika (osembitna števnika) poskrbita za ustrezne zakasnitve. Prvi rabi za vhodno-izhodno zakasnitev. Zakasnitev je potrebna za varni vklop ali izklop naprave pri zapuščanju oz. vstopanju v varovano območje, ne da bi s tem sprožili alarm. Drugo časovno vezje po določenem času izklopi zvočni signal (alarm). Za resetiranje pomnilnih celic JK uporabimo tri 4-bitne pomikalne registre, ki generirajo kratek impulzni signal. Prvi ob prekinitvi varnostne linije pošlje impulz na celico JK, ki krmili sireno, da naprava ne bi neprestano oddajala alarmnih signalov. Drugi ima nalogo, da po preteku določenega časa briše pomnilno celico JK in tako izklopi zvočno signalizacijo. Naloga zadnjega pomikalnega registra je, da napravo ob vklopu postavi v začetno stanje.

Opis signalov

Signal	Kratek opis signala
C_1	postavi se na 1, ko prvi števnik prešteje do 128
C_2	postavi se na 1, ko drugi števnik prešteje do 32
CLK	urin signal
On/Off	vklop/izklop
q_1	stanje za zakasnjeno linijo
q_2	stanje za varnostno linijo
q_3	stanje luči
q_4	stanje prikazovalnika
R_1	signal reset za brisanje JK-celic
R_2	signal za postavitev JK-celice (varnostne linije)
Reset1	resetiranje celotnega vezja (tipka reset)
Reset2	resetiranje ob izklopu
Reset3	resetiranje ob vklopu
T_1	tipalo 1
T_2	tipalo 2
T_3	tipalo 3
T_n	skupno trenutno stanje vseh tipal
V_1	varnostna linija 1
V_2	varnostna linija 2
V_3	varnostna linija 3
V_4	varnostna linija 4 – varovanje same naprave

Opis elementov

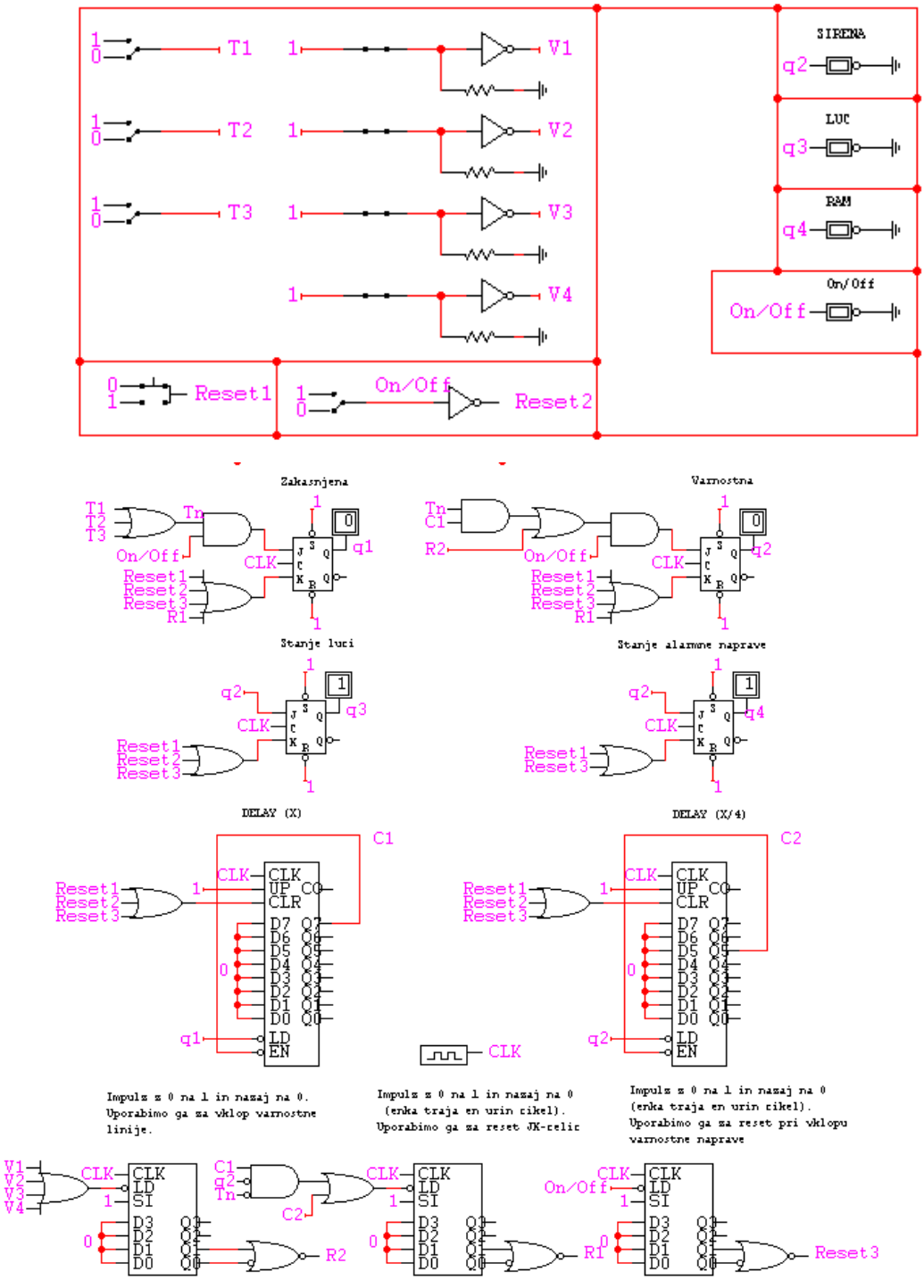
Trije štiribitni pomikalni registri:

Štiribitni pomikalni register lahko uporabimo v paralelnem ali serijskem načinu delovanja. Vrednost signala na priključku LD določa, ali pomikalni register uporabljamo za serijski (LD=1) ali paralelni (LD=0) prenos signalov. Če je vrednost signala na priključku LD na 0, se vrednosti podatkovnih vhodov D0, D1, D2 in D3 ob naslednjem urinem impulzu prenesejo na izhode Q0, Q1, Q2 in Q3. Če pa je vhodni signal LD na 1, se ob naslednjem urinem impulzu na izhod Q0 prenese vrednost signala z vhoda SI, vrednost izhoda Q0 se prenese na izhod Q1, vrednost Q1 na Q2, vrednost Q2 na Q3, stara vrednost Q3 pa se zbriše. V našem primeru uporabljamo serijski način (LD=1). Vrednost signala SI (SI=1) se ob prvem urinem impulzu prenese na izhod Q0 (Q0=1), ob vsakem naslednjem urinem impulzu pa na 1 postavi še preostale izhode Q1, Q2 in Q3.

Dva osembitna števnika:

Priključki od D0 do D7 predstavljajo začetno stanje števnika, v našem primeru so postavljeni na 0. Števnik šteje od nič navzgor, kar smo določili z vrednostjo priključka UP. Štetje se prične, ko se na priključku LD pojavi enica, ki števnik postavi v začetno stanje. Nastavimo lahko tudi zgornjo mejo štetja, do katere bo števnik štel. Priključek Q_n smo povezali s priključkom EN, kar pomeni, da ko je priključek Q_n na ena, se štetje števnika ustavi.

Vezje varnostne naprave



Slika 7.7-2 . Vezje varnostne naprave.

7.8 Projekt 9: Dvigalo za prevoz oseb

Raziskali smo različne sisteme dvigal in ugotovili, da se med seboj v glavnem razlikujejo po načinu odpiranja in zapiranja vrat. Nekatera dvigala imajo samo zunanja vrata, ki se odpirajo in zapirajo ročno ali avtomatsko, nekatera pa imajo dvojna (zunanja in notranja) vrata, od katerih se prav tako lahko ena ali obojna zapirajo ročno ali avtomatsko. Odločili smo se za izdelavo sistema, ki ima zunanja vrata, ki se odpirajo in zapirajo ročno, ter notranja, ki se zapirajo avtomatsko. Za vrata definiramo tudi vhodno in izhodno spremenljivko. Vhodna nam daje informacijo o stanju zunanjih vrat, izhodna pa krmili notranja vrata. Ker bi bilo reševanje problema v celoti prezapleteno, ga razgradimo na manjše gradnike, ki jih na koncu med seboj ustrezno povežemo. Nekateri gradniki tako predstavljajo sekvenčna, drugi kombinacijska in tretji pomnilna vezja. V nadaljevanju bomo opisali posamezne gradnike.

Opis signalov v vezju

Najprej predstavljamo vse vhodne in izhodne spremenljivke ter stanja dvigala (slika 7.8-14). Podamo jih v obliki tabel, ki prikazujejo njihova imena, opis ter binarne vrednosti.

Vhodne spremenljivke

Vhodne spremenljivke razdelimo na dva dela:

- spremenljivke, katerih vrednost generira sama naprava (tipala, senzorji ...) (tabela 7.8-1),
- spremenljivke, ki jim uporabnik spreminja vrednost s pritiskom na gumbe (tabela 7.8-2).

Tabela 7.8-1. Pregled vhodnih spremenljivk, ki jih krmilijo zunanje strojne naprave.

Ime	Vrednost	Opis
Vin	Vrednost se postavi na 1, če so odprta vrata v vsaj enem nadstropju, sicer je vrednost enaka 0.	indikator vrat v nadstropjih
OverLoad	Vrednost se postavi na 1, če je dvigalo preobremenjeno, sicer je vrednost enaka 0.	indikator preobremenjenosti dvigala

Tabela 7.8-2. Pregled vhodnih spremenljivk, ki jih uporabnik krmili s pritiski na gumbe.

Ime	Vrednost	Opis
2Td	Vse spremenljivke se spreminjajo na enak način: Ko uporabnik pritisne tipko, se vrednost postavi na 1, takoj ko jo izpusti, pa se postavi nazaj na 0.	tipka za klic dvigala v 2. nadstropju
1Tg		tipka za klic dvigala v 1. nadstropju, če se želimo peljati navzgor
1Td		tipka za klic dvigala v 1. nadstropju, če se želimo peljati navzdol
PTg		tipka za klic dvigala v pritličju, če se želimo peljati navzgor
PTd		tipka za klic dvigala v pritličju, če se želimo peljati navzdol
KTg		tipka za klic dvigala v kleti
2		tipka v dvigalu, s katero povemo, da želimo v 2. nadstropje
1		tipka v dvigalu, s katero povemo, da želimo v 1. nadstropje
P		tipka v dvigalu, s katero povemo, da želimo v pritličje
K		tipka v dvigalu, s katero povemo, da želimo v klet
Pomoc		tipka v dvigalu, s katero ustavimo dvigalo in pokličemo hišnika
Rn		tipka, s katero hišnik izklopi indikator napake, potem ko jo odpravi

Izhodne spremenljivke

Tudi izhodne spremenljivke razdelimo na dva dela. Z enimi sistem preko lučk uporabnika obvešča o stanju dvigala (tabela 7.8-3), druge pa potrebujemo za krmiljenje zunanjih strojnih naprav (tabela 7.8-4).

Tabela 7.8-3. Pregled izhodnih spremenljivk, ki uporabnika obveščajo o stanju dvigala.

Ime	Vrednost	Opis
N0...N7	ASCII Dec. Bin. K 75 01001011 P 80 01010000 1 49 00110001 2 50 00110010	osembitna beseda, ki ASCII prikazovalniku pove, katere znake naj izpisuje
N	1 – v sistemu je napaka 0 – sistem deluje pravilno	prižge lučki v dvigalu in pri hišniku, ki javljata napako

Tabela 7.8-4. Pregled izhodnih spremenljivk, ki krmilijo zunanje strojne naprave.

Ime	Vrednost	Opis
S1, S2	0 0 stoj 0 1 poženi navzgor 1 0 poženi navzdol 1 1 nedefinirano stanje (ne napravi ničesar)	spremenljivki upravljata motor, ki poganja dvigalo
Vout	0 – vrata zaprta, 1 – vrata odprta	krmiljenje vrat v dvigalu

Opis pozicije dvigala

S spremenljivkami sistema lahko podamo tudi nadstropje, v katerem se dvigalo nahaja. Kako so kodirana ta stanja, je opisano v tabeli 7.8-5.

Tabela 7.8-5. Oznake nadstropij.

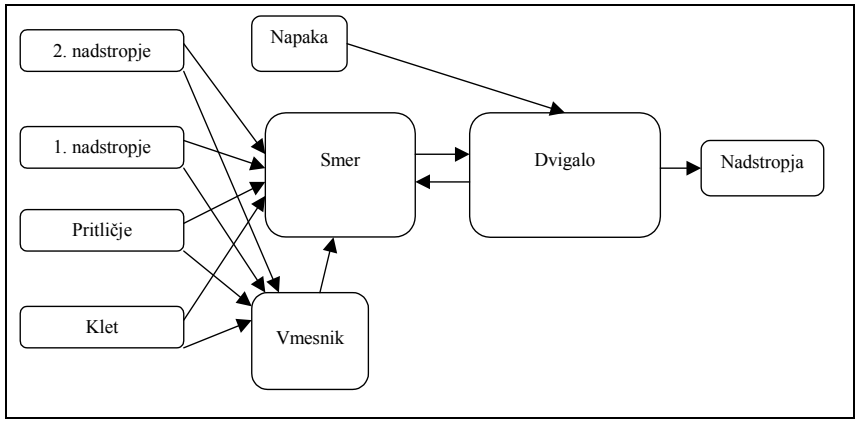
Nadstropje	Kratka oznaka	X_1	X_2
Klet	K	0	0
Pritličje	P	0	1
1. nadstropje	1	1	0
2. nadstropje	2	1	1

Blokovna shema avtomata

Sistem razdelimo na naslednje bloke (slika 7.8-13):

- Napaka,
- Dvigalo,
- Nadstropja,
- Smer,
- Vmesnik,
- Klet,
- Pritličje,
- 1. nadstropje,
- 2. nadstropje.

Bloki med seboj komunicirajo, kot je prikazano na sliki 7.8-1, v nadaljevanju pa opišemo tudi njihovo delovanje.



Slika 7.8-1. Komunikacija med posameznimi bloki.

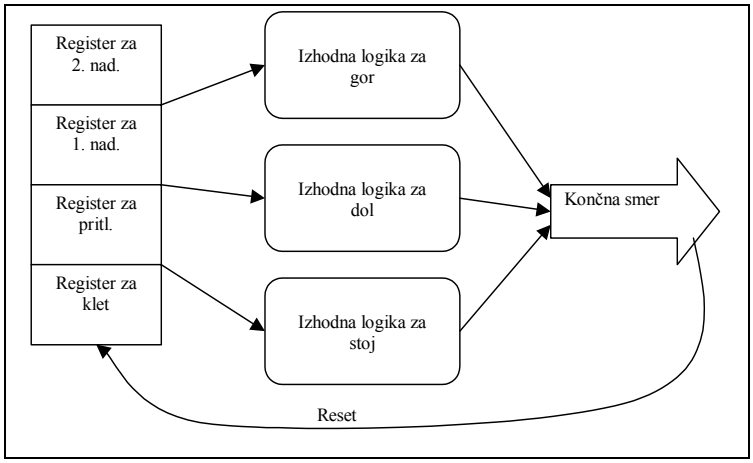
Kratek opis modulov vezja

Kontrolniki nadstropij

Za vsako nadstropje izdelamo svoj kontrolnik, ki glede na pritisnjeno tipko, stanje in smer dvigala odloči, ali bo zahteva takoj poslana v modul **`'Smer'`** ali pa bo najprej prenesena v **`'Vmesnik'`**.

Smer

Ta modul daje modulu **`'Dvigalo'`** napotke, v katero smer naj nadaljuje vožnjo. Sestavljen je iz registra zahtev, ki jih izvršuje po najbolj optimalnem vrstnem redu, in kombinacijske logike. Modul je predstavljen na sliki 7.8-2.



Slika 7.8-2. Shematski prikaz pretoka podatkov v modulu **`'Smer'`**.

Napaka

Ta modul odkriva napake v sistemu in jih javlja modulu **'Dvigalo'**. Do napake lahko pride zaradi odprtih vrat, preobremenitve dvigala ali ker potnik pritisne tipko **'Pomoč'**. Signal za napako je prisoten tako dolgo, dokler se vrata ne zaprejo, dvigalo ni več preobremenjeno ali dokler serviser ne pritisne tipke **'Reset'**.

Dvigalo

Modul **'Dvigalo'** je osrednji del sistema. Je nosilec informacije o stanju (nadstropju), v katerem se dvigalo trenutno nahaja, in ga posreduje modulom, ki ta podatek potrebujejo. Stanje spreminja modul **'Smer'**, ki lahko premakne dvigalo eno nadstropje višje ali nižje.

Nadstropja

Ta modul je preprosto kombinacijsko vezje, ki pretvori dvobitni zapis trenutnega nadstropja v osembitnega in ga posreduje zaslonu ASCII, ki ga izpiše.

Vmesnik

Je štiribitni pomnilnik, na katerega prihajajo vhodi (zahteve, ki čakajo na izvršitev), ki si jih zapomni in jih na poziv, ki ga dobi iz modula **'Smer'**, le-temu posreduje in se nato resetira ter čaka na nove vhode.

Realizacija podvezij

Kontrolniki nadstropij

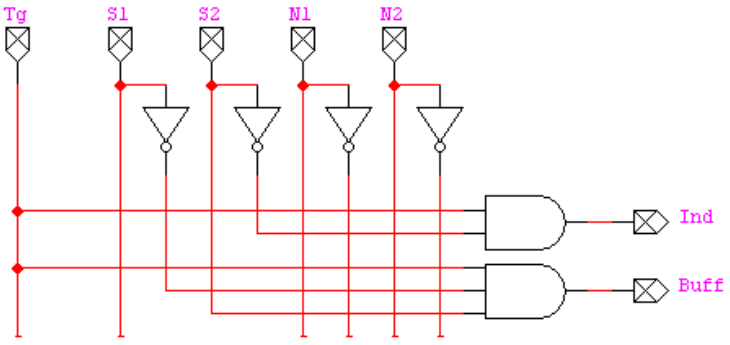
```
Klet.cct, Pritl.cct, 1nad.cct, 2nad.cct
```

Za vsako nadstropje sta namenjeni največ dve vhodni spremenljivki kot gumba na komandni plošči (npr. **Ptg, Ptd**) ter spremenljivke, ki določajo trenutno pozicijo in smer dvigala. Izhodni spremenljivki sta **Ind**, ki modulu **'Smer'** pove, naj se ustavi v tem nadstropju, ter **Buff**, katere vrednost se zapiše v **'Vmesnik'**. Oba izhoda se prožita ob pritisku gumba, vendar drug drugega izključujeta. Kontrolnike nadstropij izdelamo tako, da definiramo vrednosti obeh izhodnih spremenljivk s pomožnimi spremenljivkami **Gor, Dol** in **Stop**, ki so v vezjih izražene s S1 in S2 (tabela 7.8-4), ter z oznakami nadstropij (tabela 7.8-5). Spremenljivki **Ind** in **Buff** sta definirani v tabeli 7.8-6. Poglejmo, kateri pogoji morajo biti izpolnjeni, da se bo signal **Ind** v vezju **2nad.cct** (slika 7.8-5) postavil na 1: v 2.

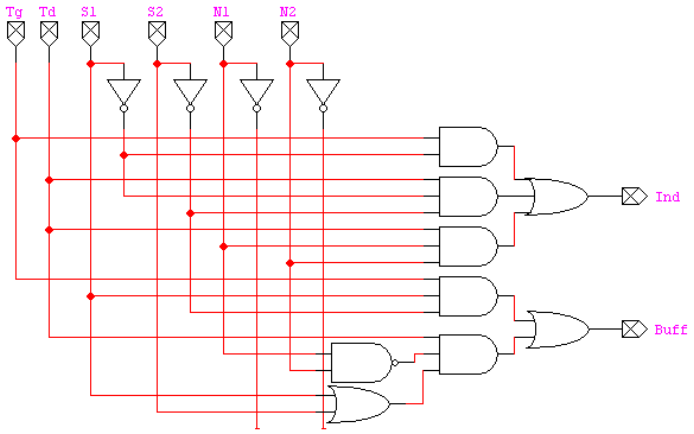
nadstropju je pritisnjena tipka **Td** in dvigalo ali potuje navzgor ali pa stoji. Da se bo v vezju 2nad.cct na 1 postavil signal **Buff**, pa morata biti izpolnjena naslednja pogoja: v 2. nadstropju je pritisnjena tipka **Td** in dvigalo potuje navzdol. V tem primeru se signal zapiše v **'Vmesnik'** in aktivnost se izvede, ko dvigalo pride v ustrezno nadstropje. Modul kontrolnika klet.cct je predstavljen na sliki 7.8-3, modul 1nad.cct na sliki 7.8-4 ter modul pritličje.cct na sliki 7.8-6.

Tabela 7.8-6. Definicija izhodnih spremenljivk v kontrolnikih nadstropij s pomočjo pomožnih spremenljivk Gor, Dol in Stop.

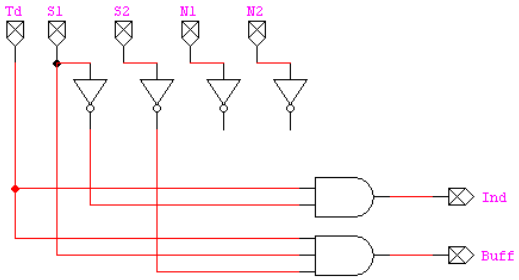
Nadstropje	Formule
2. nadstropje	Ind := Td (Gor + Stop);
	Buff := Td Dol;
1. nadstropje	Ind := Tg (Gor + Stop) + Td (2 + Stop);
	Buff := Tg Dol + Td 2';
Pritličje	Ind := Tg (K + Stop) + Td (Dol + Stop);
	Buff := Tg K' + Td Gor;
Klet	Ind := Tg (Dol + Stop);
	Buff := Tg Gor;



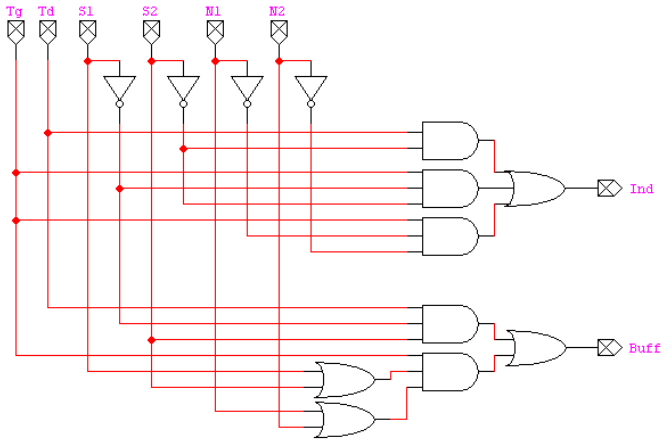
Slika 7.8-3. Vezje klet.cct.



Slika 7.8-4. Vezje 1nad.cct.



Slika 7.8-5. Vezje 2nad.cct.



Slika 7.8-6. Vezje pritličje.cct.

Vezje Smer (smer.cct)

Če predstavlja modul **'Dvigalo'** »srce« sistema, potem predstavlja modul **'Smer'** »možgane« sistema (slika 7.8-7). Na vhodnih priključkih dobi zastavico za posamezna nadstropja, pomni njihova stanja, nato pa dvigalo vodi od nadstropja do nadstropja in ko pride do takšnega z zastavico, se ustavi in le-tega zbriše.

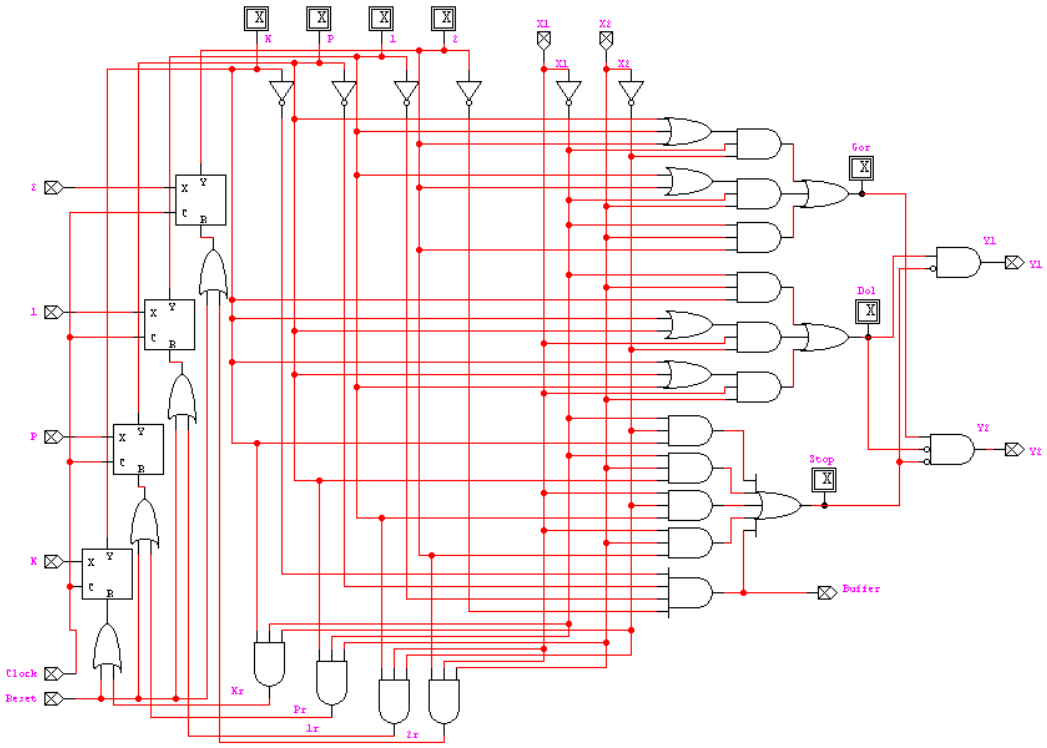
Kako vezje odloča, v katero smer bo dvigalo nadaljevalo pot glede na te zastavice (signal na 1 na enem od vhodov **K, P, 1, 2** vezja smer.cct) in trenutni položaj dvigala (kombinacija signalov **x1x2** (slika 7.8-7) oz. **s1s2** (slika 7.8-13)), določimo s psevdokodom na podoben način kot v tabeli 7.8-6.

```
Gor := S1'S2' (P + 1 + 2) + S1'S2 (1 + 2) + S1S2' 2;

Dol := S1'S2 K + S1S2' (K + P) + S1S2 (K + P + 1);

Stop := (S1'S2' K) + (S1'S2 P) + (S1S2' 1) + (S1S2 2);
```

Tako smo dobili tri pomožne spremenljivke (**Gor**, **Dol**, **Stop**). Ker obstaja teoretična možnost, da se nekatere akcije aktivirajo istočasno, smo napravili še izhodno logiko, ki poda končno smer. Ta je določena v tabeli 7.8-7.



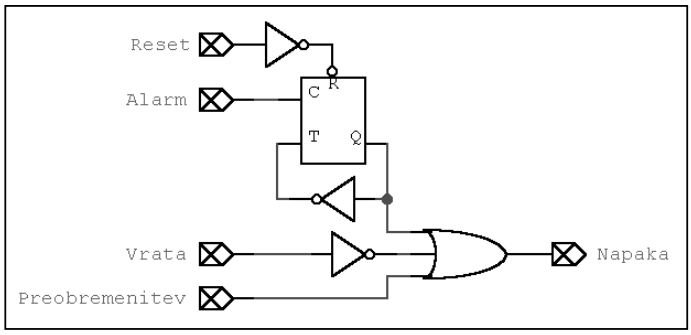
Slika 7.8-7. Vezje smer.cct.

Tabela 7.8-7. Pravilnostna tabela za izhodno logiko v modulu **`'Smer'`.**

Gor	Dol	Stop	Y1	Y2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	1	0
1	1	1	0	0

Vezje Napaka (napaka.cct)

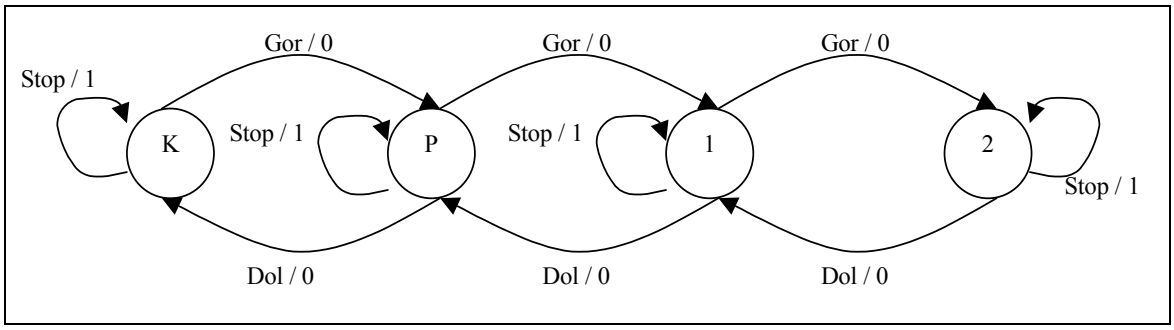
Vezje za javljanje napake je relativno preprosto (slika 7.8-8). Za vhodne spremenljivke rabijo senzor za preobremenitev, senzor za odprta vrata in gumb **`'Pomoč'`** (brišemo ga le z gumbom **`'Reset'`**), na izhodu pa vezje da signal za napako.



Slika 7.8-8. Vezje za javljanje napake.

Vezje Dvigalo (dvigalo.cct)

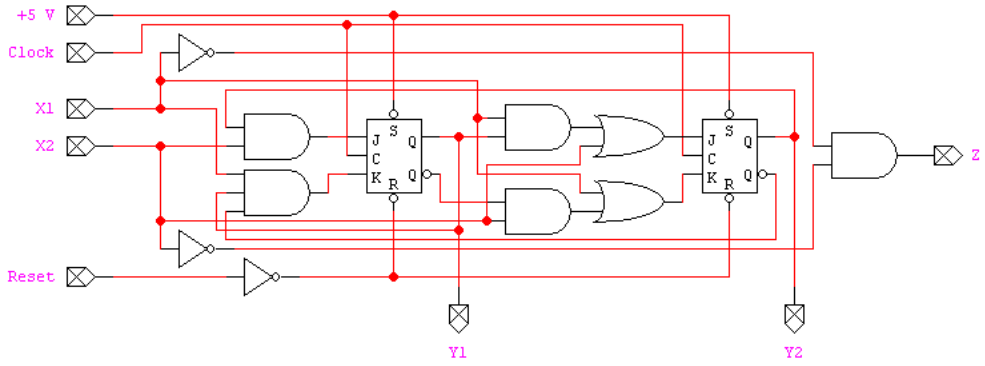
Modul **`'dvigalo'`** realiziramo s sinhronim sekvenčim vezjem (slika 7.8-9), ki ima štiri stanja (**`K, P, 1, 2`**), prehode med stanji pa določamo s smerjo dvigala.



Slika 7.8-9. Diagram prehajanja stanj (nadstropij) dvigala.

V diagramu so podani pogoji za prehod iz enega v drugo stanje z **`Gor`**, **`Dol`** in **`Stop`**, ki so kodirani v tabeli 7.8–4. Izhodna spremenljivka pa je spremenljivka **`Vout`**.

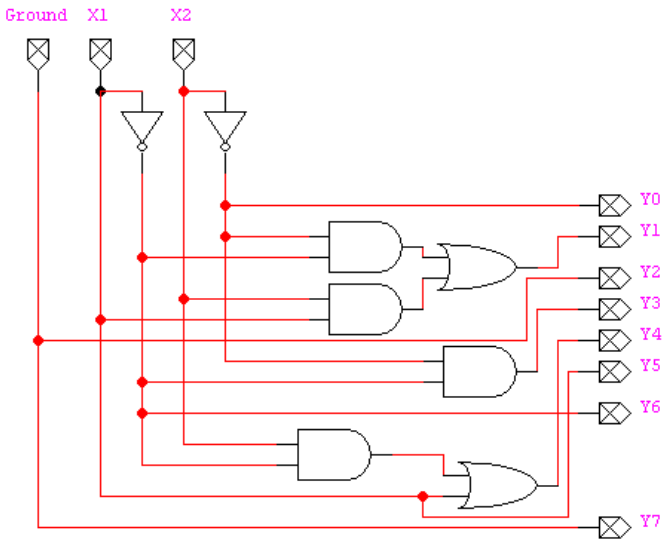
Za realizacijo tega modula uporabimo dve pomnilni celici JK. Diagram pretvorimo v tabelo in realiziramo vezje (slika 7.8-10).



Slika 7.8-10. Modul dvigalo.cct.

Nadstropja (nadstropja.cct)

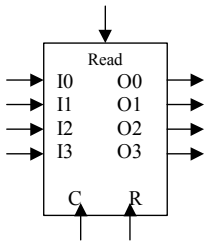
Modul **`'Nadstropja'`** je preprosto kombinacijsko vezje (slika 7.8-11). Vezje štirim kombinacijam dveh bitov enolično priredi štiri kombinacije osmih bitov.



Slika 7.8-11. Vezje nadstropja.cct.

Vmesnik (vmesnik.cct)

Modul **'Vmesnik'** si zapomni vrednosti signalov na vhodnih priključkih I0 do I3. Signal Read aktivira prenos vrednosti na izhodne priključke vmesnika O0 do O3. Signal **'Reset'** nato vrednosti posameznih celic bufferja postavi na 0. Na priključek **'C'** priključimo urin signal. Slika 7.8-12 prikazuje priključno shemo modula **'Vmesnik'**.



Slika 7.8-12. Modul **'Vmesnik'**

Povezava posameznih modulov (main.cct)

Main.cct

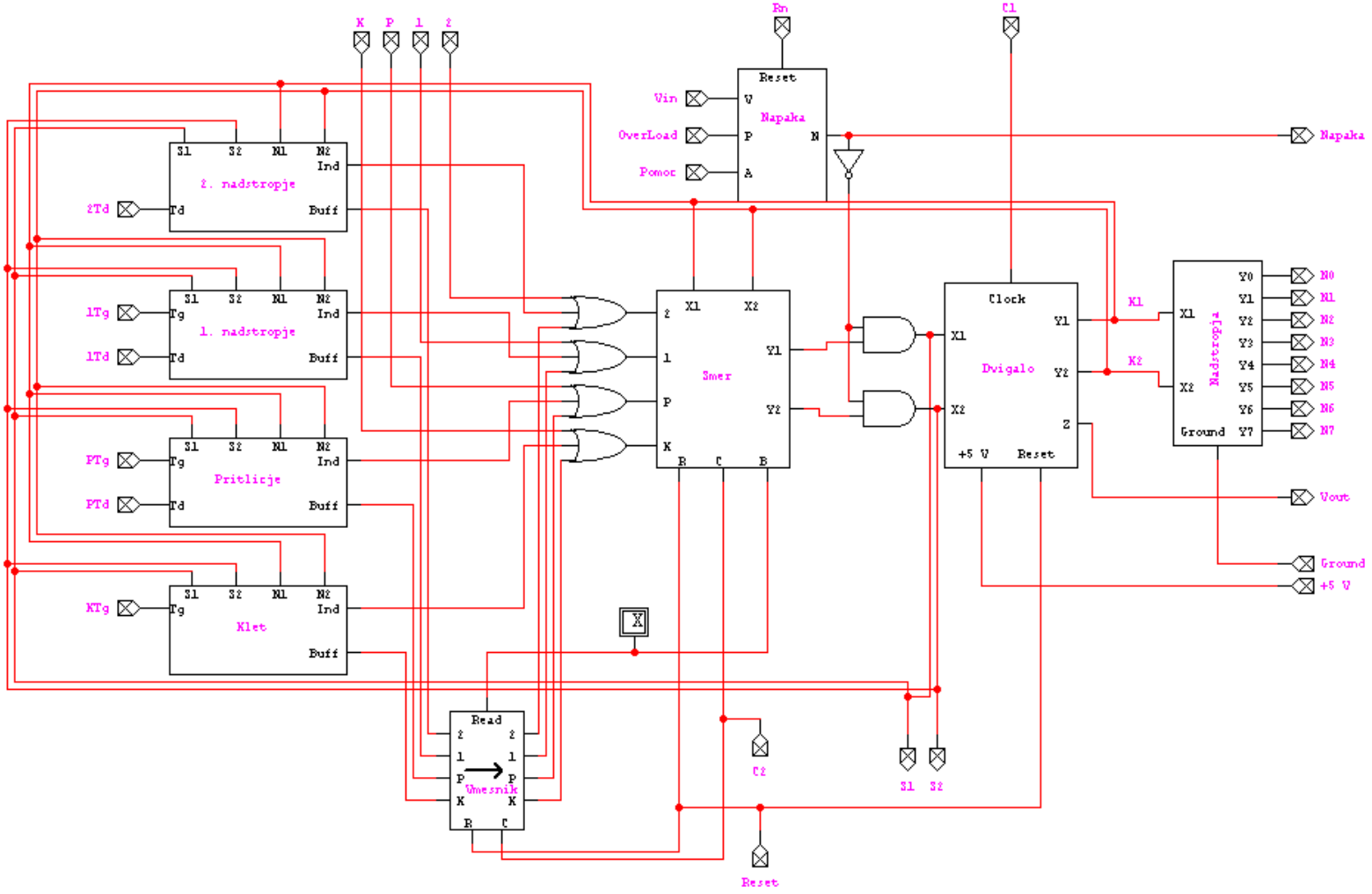
Moduli medsebojno komunicirajo s signali (slika 7.8-13); iz modulov **'Klet', 'Pritl', '1nad'** in **'2nad'** gredo povezave po imenu **'Ind'** do priključkov **K, P, 1** in **2** na modulu **'Smer'**, ter povezave **'Buff'** na istoimenske priključke na modulu **'Vmesnik'**. Iz modula **'Smer'** gresta **S1** in **S2** v modul **'Dvigalo'** ter v module po posameznih nadstropjih. Izhodi **K, P, 1** in **2** iz modula **' Vmesnik'** gredo tudi na istoimenske vhode

modula **'Smer'**. Modul **'Napaka'** da svoj edini signal v kombinaciji s **S1** in **S2** iz modula **'Smer'** modulu **'Dvigalo'**. Modul **'Dvigalo'** javlja stanje dvigala preko **K1** in **K2** modulom **'Nadstropja'**, **'Smer'** ter modulom posameznih nadstropij.

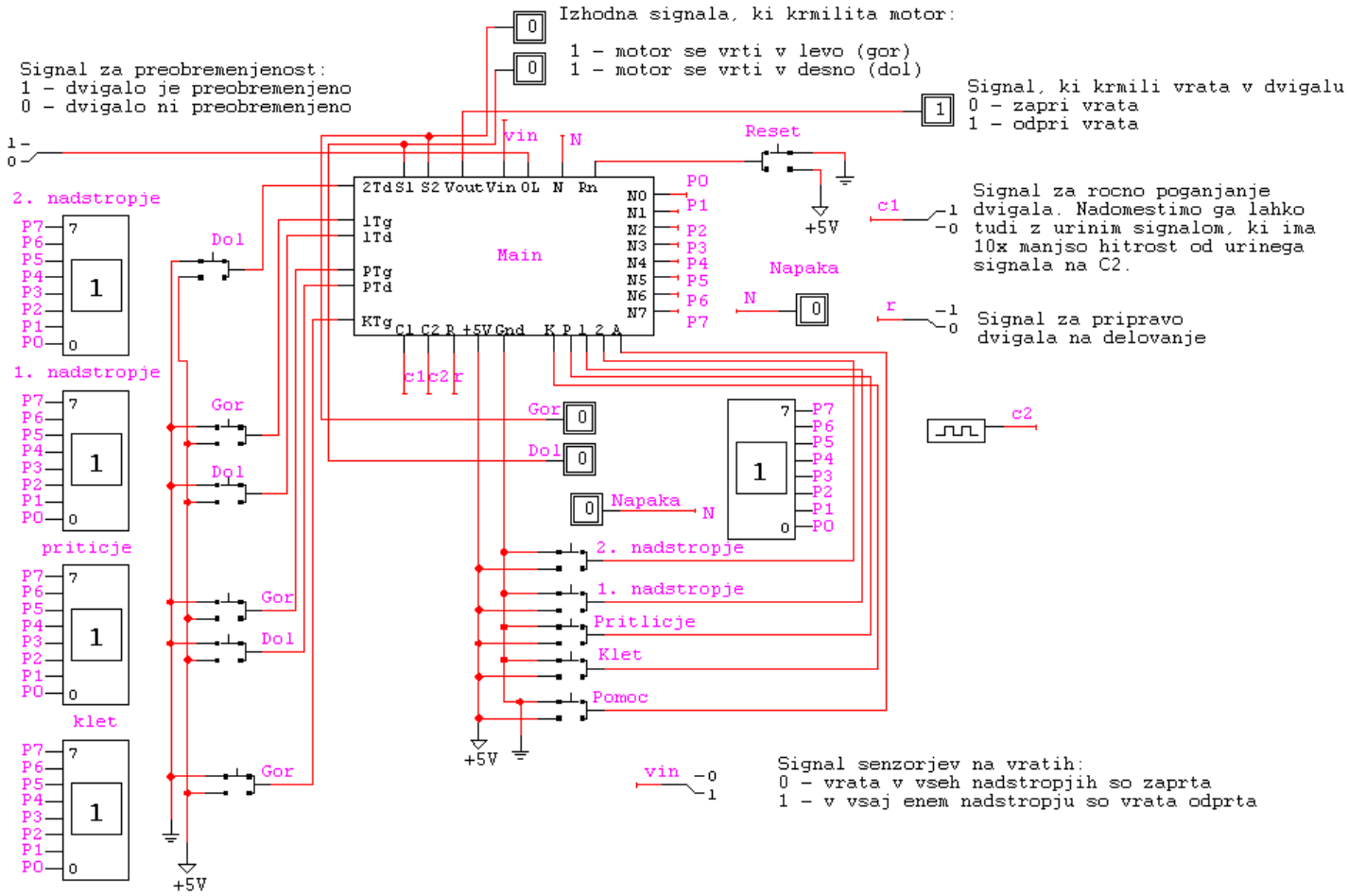
Priključitev zunanjih vhodov in izhodov (test.cct)

Test.cct

Za enostavnejši prikaz delovanja vezja dvigala smo izdelali element **'Main'**(slika 7.8-13). Priključitev zunanjih vhodov in izhodov smo prikazali v datoteki **Test.cct** (slika 7.8-14), kjer smo priklopili vse zunanje signale, da lahko opazujemo delovanje vezja. Za gumbe za klic dvigala v posameznih nadstropjih in v dvigalu samem smo uporabili elemente iz knjižnic **'SPDT PushButton'**, ker nam omogočajo simulacijo pritiska gumbov na plošči z enojnim klikom miške. Namesto svetlečih lučk smo uporabili elemente **'Binary probe'.** Te elemente smo uporabili tudi za priklop zunanjih strojnih naprav.

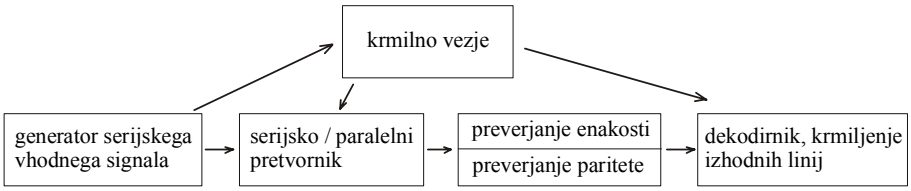


Slika 7.8-13. Podvezje dvigala Main.



Slika 7.8-14. Celotno vezje dvigala.

7.9 Projekt 12: Sprejemnik in dekodirnik signala



Slika 7.9-1. Blokovna shema vezja.

Opis signalov

Opis vhodnega signala

Za dekodiranje vhodnega signala moramo poznati zgradbo zaporedja vhodnega signala. V tabeli določimo pomen posameznih vhodnih bitov. Vhodni signal je sestavljen iz enega začetnega bita, štirih podatkovnih bitov in paritetnega bita.

0000...00001 $X_1X_2Y_1Y_2$ P00000...00000

paritetni bit

podatkovni biti

začetni bit

Z začetnim bitom določimo začetek podatkovnega bloka, s podatkovnimi biti pa predstavimo vsebino sporočila. Sestavljeni so iz dveh enakih delov (prvi del vsebuje bita X_1X_2, drugi pa Y_1Y_2). Pošiljanje podvojenih podatkovnih bitov je izvedeno zaradi preverjanja pravilnosti. Tudi paritetni bit je namenjen preverjanju pravilnosti pri prenosu.

Opis signalov v vezju

Inicializacijski signal je namenjen postavitvi krmilnega vezja v začetno stanje. To je signal, ki aktivira števnik, kateri šteje prebrane podatkovne bite. Ko krmilno vezje ugotovi, da je prispel paritetni bit, se s pomočjo inicializacijskega signala vrne v začetno stanje, kjer spet čaka na naslednji začetni bit.

Krmilni signal za krmiljenje serijsko/paralelnega pretvornika se nastavi na 0, da pretvornik paritetnega bita ne prebere, saj bi bilo preverjanje pravilnosti sicer napačno. *Krmilni signal za*

omogočitev izhodnih linij nastavi izhodne linije na 1, ko so prebrani vsi podatkovni biti in pariteta. Takrat se preveri, ali je prenos bil pravilen, da bi lahko sprostili izhodne linije. *Kontrolni signal enakosti* je na 1, kadar sta oba dela podatkovnih bitov, X_1X_2 in Y_1Y_2, enaka.

Kontrolni signal paritete je na 1, če da kombinacija signalov P in Y_1 Y_2 liho število enic.

Izhodne signale dajejo 4 vzporedne povezave. Vsaka lahko upravlja po eno napravo. Krmilnik izhodnih povezav je napravljen tako, da se izbrana povezava postavi v aktivno stanje (izhod je na 1) in v tem stanju ostane do naslednje izbire izhodne povezave.

Preverjanje pravilnosti

Uporabljamo dva načina preverjanja pravilnosti. Prvi način je preverjanje enakosti, drugi pa paritete. Z dvojnim preverjanjem prepoznamo nepravilen vhodni signal kot pravilnega le v primeru, če se pri prenosu invertirajo vsi 4 podatkovni biti.

Pri preverjanju enakosti ugotavljamo, ali je prvi par podatkovnih bitov X_1X_2 enak drugemu, Y_1Y_2. Če se par X_1X_2 razlikuje od para Y_1Y_2, je prišlo do napake pri prenosu in sprejem na izhodu nima vpliva. Če pa se tako par X_1X_2 kot par Y_1Y_2 spremenita in sta po naključju ponovno enaka, napako lahko ugotovi preverjanje paritete.

Preverjanje paritete pa deluje na znani način; povedati moramo le to, da je upoštevana liha pariteta in da se preverjanje izvrši na paru Y_1Y_2.

Diagrami prehajanja stanj, tabele, minimizacije in enačbe

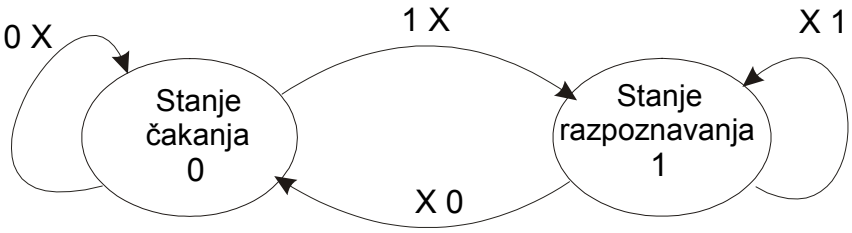
Namen vezja je, da razpozna serijski vhodni signal in pretvori le-tega v paralelne izhodne signale, ki krmilijo ustrezne naprave. V tabeli 7.9-1 je prikazano, katera od naprav a, b, c ali d bo izbrana v odvisnosti od kombinacije signalov X_1X_2.

Tabela 7.9-1. Kombinacija signalov X_1X_2 izbere izhodno napravo a,b c ali d.

Vhod		Izhod			
X_1	X_2	a	b	c	d
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Da bi bili na vhodu vezja signali pravilno razpoznani, moramo ugotoviti, kdaj v vezje prihajajo podatkovni biti. Začetek podatkovnih bitov ugotavljamo s preprostim sekvenčnim vezjem, ki

ima dve stanji; v prvem čaka na začetni bit, v drugem pa bere in razpoznava podatkovne bite. Omenjeno sekvenčno vezje podajamo z diagramom prehajanja stanj (slika 7.9-2). Prehode med stanjema 0 in 1 določa kombinacija vhodnega in krmilnega signala S/P pretvornika.



Slika 7.9-2. Diagram prehajanja stanj.

Ker mora biti vezje v stanju razpoznavanja štiri urine periode (štirje podatkovni biti), potrebujemo vezje, ki bo štelo urine periode in po štirih nastavilo krmilni signal za krmiljenje S/P pretvornika na 0:

števnik	krmilni signal za omogočitev izhodnih linij	krmilni signal za krmiljenje S/P pretvornika
0	0	1
1	0	1
2	0	1
3	0	1
4	0	0
5	1	1

Pravilnost vhodnega signala najprej ugotavljamo s preverjanjem enakosti. V tabeli 7.9-2 primerjamo, ali sta vhodna signala X_1 X_2 in Y_1 Y_2 enaka. Če sta, je kontrolni signal na 1. Nato pa preverimo še pariteto. Uporabljena je liha pariteta. Kontrolni signal za pariteto je na 1, če da kombinacija signalov P in Y_1 Y_2 liho število enic (tabela 7.9-3).

Tabela 7.9-2. Primerjalna tabela kombinacij vhodnega signala X_1 X_2 Y_1 Y_2.

X_1	X_2	Y_1	Y_2	kontrolni signal enakost
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

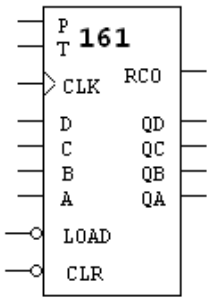
Tabela 7.9-3. Kontrola paritete.

pariteta (P)	Y_1	Y_2	kontrolni signal paritete
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Opis uporabljenih elementov in njihovih funkcij

4-bitni binarni sinhroni števnik LS161

Na sliki 7.9-3 je shema vezja števnika LS161, v tabeli 7.9-4 je opis posameznih priključkov elementa in v tabeli 7.9-5 opis njegovega obnašanja.



Slika 7.9-3. 4-bitni binarni sinhroni števnik LS161.

Tabela 7.9-4. Tabela z opisom priključkov 4-bitnega binarnega sinhronega števnika LS161.

Priključek	Opis priključka
CLK	Vhod za urin signal. Vhod je aktiven na pozitivno fronto.
LOAD	Če je priključek na 0, omogoča pričetek štetja števnika od vrednosti, ki jo nastavimo na priključkih od A do D.
CLR	Če je priključek na 0, se izvede asinhrono brisanje števnika (izhodi od QA do QD se postavijo na 0).
P, T	Omogočitvena vhoda omogočita štetje. Eden od njiju je lahko uporabljen skupaj z izhodom RCO predhodnega števnika. Na tak način lahko enostavno naredimo N-bitni števnik.
A, B, C, D	Priključki, na katerih nastavimo začetno vrednost štetja.
QA, QB, QC, QD	Izhodni priključki števnika.

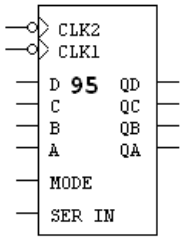
Tabela 7.9-5. Delovanje števnika 161 in 163.

M54/74HCT160/161					M54/74HCT162/163					OUTPUTS				FUNCTION
INPUTS					INPUTS									
$\overline{CLR}$	$\overline{LD}$	PE	TE	CK	$\overline{CLR}$	$\overline{LD}$	PE	TE	CK	QA	QB	QC	QD	
L	X	X	X	X	L	X	X	X	⎍	L	L	L	L	RESET TO "0"
H	L	X	X	⎍	H	L	X	X	⎍	A	B	C	D	PRESET DATA
H	H	X	L	⎍	H	H	X	L	⎍	NO CHANGE				NO COUNT
H	H	L	X	⎍	H	H	L	X	⎍	NO CHANGE				NO COUNT
H	H	H	H	⎍	H	H	H	H	⎍	COUNT UP				COUNT
H	X	X	X	⎎	X	X	X	X	⎎	NO CHANGE				NO COUNT

Note: X : Don't Care
A, B, C, D : Logi level of data inputs
Carry : CARRY = TE • Q_A • $\overline{Q_B}$ • $\overline{Q_C}$ • Q_D ..... (M54/74HCT160/162)
: CARRY = TE • Q_A • Q_B • Q_C • Q_D ..... (M54/74HCT161/163)

4-vhodni serijsko/paralelni pomikalni register

Na sliki 7.9-4 je shema vezja serijsko/paralelnega pomikalnega registra LS95, v tabeli 7.9-6 pa je opisano obnašanje pomikalnega registra in opis posameznih priključkov.



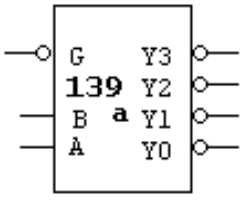
Slika 7.9-4. Serijsko/ paralelni pomikalni register

Tabela 7.9-6. Funkcijska tabela štirivhodnega serijsko/paralelnega pomikalnega registra LS95.

Vhodni priključki					Izhodni priključki			
CLK1	CLK2	MODE	SER IN	Paralelni način A B C D	QA	QB	QC	QD
0-1	0-1	1	X	a b c d	a	b	c	d
0	0	X	X	X X X X	ni sprememb			
0-1	X	0	0	X X X X	0	QA	QB	QC
0-1	X	0	1	X X X X	1	QA	QB	QC

Dekodirnik/demultipleksor 1 na 4

Na sliki 7.9-5 je shema vezja dekodirnika/demultipleksorja LS139, tabela 7.9-7 pa je funkcijska tabela vhodno/izhodnih priključkov.



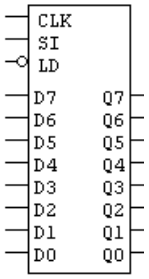
Slika 7.9-5. Vezalna shema dekodirnika/demultipleksorja LS139.

Tabela 7.9-7. Funkcijska tabela dekodirnika/demultipleksorja LS139.

Vhodni priključki			Izhodni priključki			
G	A	B	Y0	Y1	Y2	Y3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Osemvhodni serijsko/paralelni pomikalni register

Na sliki 7.9-6 je shema vezja serijsko/paralelnega pomikalnega registra (knjižnica LW3), v tabeli 7.9-8 pa so opisani priključki elementa.



Slika 7.9-6.

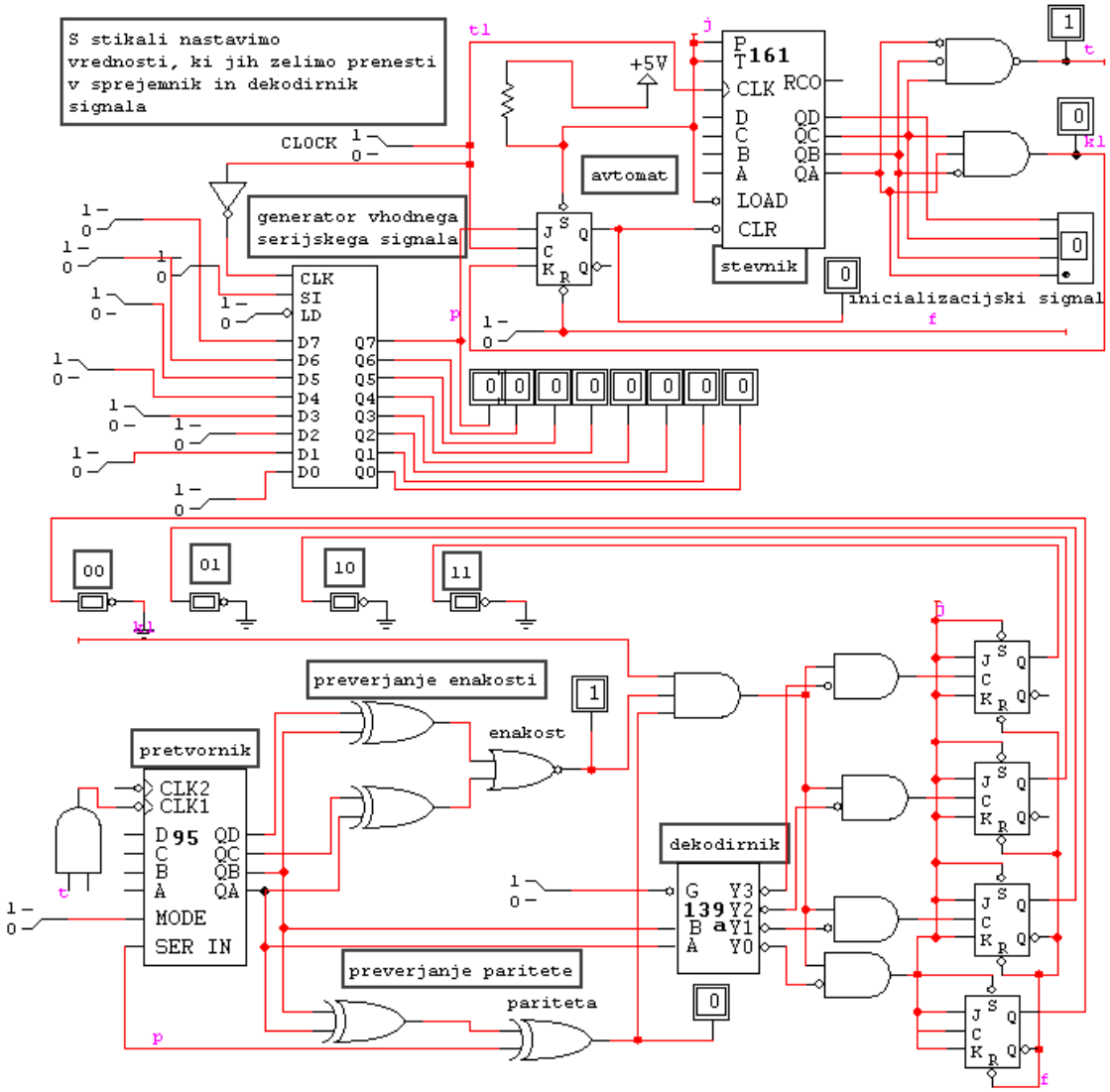
Serijsko/ paralelni pomikalni register

Tabela 7.9-8. Opis priključkov osemvhodnega serijsko/paralelnega pomikalnega registra.

Priključki	Opis in pomen priključka
CLK	Pri vsakem prehodu urinega signala z 0 na 1 se preberejo vrednosti signalov na vhodnih priključkih od D0 do D7 oz. na priključku SI. Pri vsakem prehodu urinega signala z 1 na 0 se prenesejo vrednosti, prebrane na priključkih od D0 do D7, na izhodne priključke od Q0 do Q7 pri izbranem paralelnem načinu oz. vrednost z SI na Q0, če je izbran serijski način prenosa.
LD = 1	Izbran je serijski prenos podatkov. Prenos podatkov poteka v smeri od izhodnega priključka Q0 v Q1, nato v Q2, ..., v Q7. Ob vsakem naslednjem prenosu izhod Q0 prebere podatek na vhodu SI.
LD = 0	Izbran je paralelni prenos podatkov. Ob prehodu urinega signala z 0 na 1 se na vhodih D0-D7 preberejo vrednosti signalov in ob prehodu urinega signala z 1 na 0 se prebrane vrednosti prenesejo na izhodne priključke Q0-Q7.
SI	Vhodni serijski priključek
D0-D7	Vhodni paralelni priključki
Q0-Q7	Izhodni priključki

Zaključek

Zgradba našega sprejemno-dekodirnega vezja (slika 7.9-7) je preprosta, vendar svojo nalogo dobro opravlja. Vezje, ki bi se tudi v praksi lahko uporabljalo, bi morda imelo še nekaj dodatnih elementov in funkcij. Če bi prenašali večje število bitov, bi lahko dodatno vgradili tudi nekatere druge mehanizme za preverjanje pravilnosti prenosa, npr. CRC.



Slika 7.9-7. Celotno vezje sprejemnika in dekodirnika signala.

7.10 Projekt 13: Inkrementalni dajalnik

Inkrementalni dajalnik je vezje, ki s pomočjo senzorjev ugotavlja smer vrtenja osi. Na vreteno osi je nameščena plošča s črnobelim vzorcem (slika 7.10-1). Ko se plošča vrti na vretenu osi, prekinja snope svetlobe fiksno nameščenih senzorjev, ki so med seboj razmaknjeni za neko razdaljo. Oddaljenost med senzorji se zmanjšuje s povečevanjem števila črnobelih polj na plošči.

Za računalniško simulacijo potrebujemo vezje, ki simulira vrtenje vretena. Potrebujemo generatorje vhodnih sekvenc, ki nadomeščajo izhodne signale senzorjev.

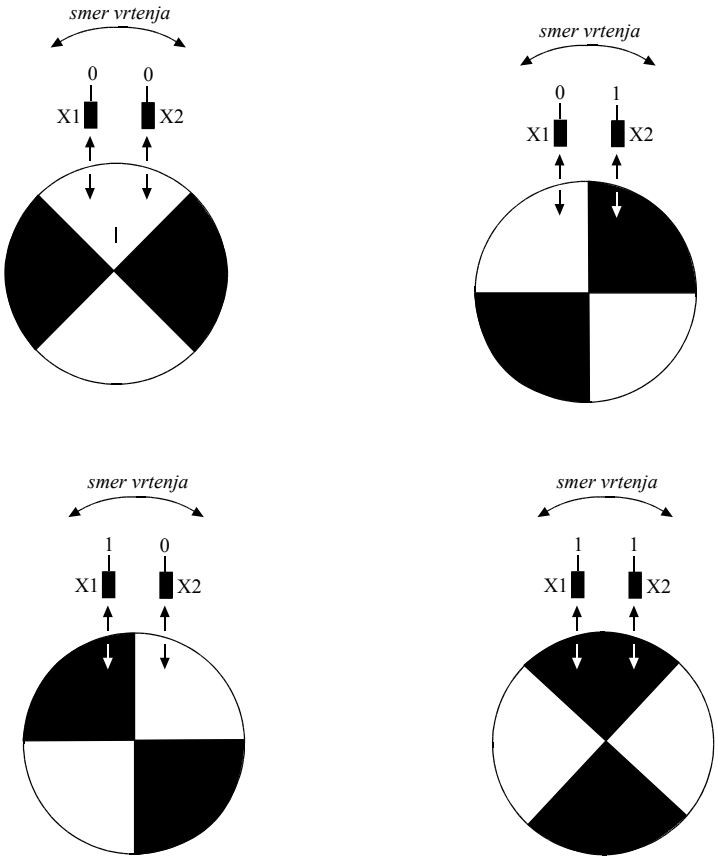
Sestavimo Mealyjev avtomat, ki ima po dva vhoda in izhoda:

Če simuliramo vrtenje v desno, na izhodih dobimo sekvenco 00, 10, 00 po polovičnem zasuku vretena v desno in sekvenco 00, 01, 00, če se vreteno zasuče za polovico v levo.

Diagram prehajanja stanj

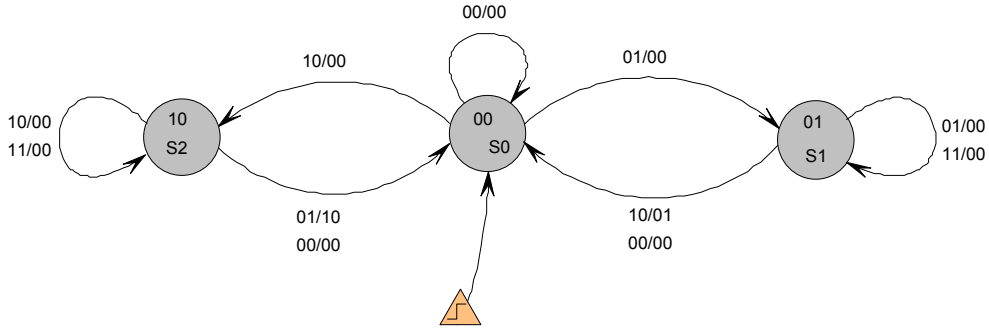
Ob resetu se vezje postavi v izhodiščno stanje S_0 (slika 7.10-2). Vhodne in izhodne vrednosti signalov so na 0. Če se na vhodih vezja X_1X_2 pojavi kombinacija 0 1, gre vezje ob naslednjem urinem signalu v stanje S_1, vrednosti obeh izhodnih signalov pa sta še vedno na 0. V tem stanju ostanemo, vse dokler sta vrednosti signalov na vhodu 0 1 in še tudi potem, ko sta vrednosti vhodnih signalov že na 1 1. Šele ko se pojavi kombinacija vrednosti signalov 1 0, vemo, da se je vreteno dejansko zasukalo za polovico obrata (v levo) in drugi izhod gre na 1 za dolžino ene periode urinega signala. Sekvenčno vezje pa je ponovno v izhodiščnem stanju S_0. Če se v stanju S_1 pojavi kombinacija vhodnih signalov 0 0, pomeni, da zasuka za pol obrata ni bilo in vezje se vrne v izhodiščno stanje S_0, izhoda pa sta oba na 0.

Postavimo vezje ponovno v izhodiščno stanje S_0, izhoda sta na 0. Če na vhode pripeljemo kombinacijo 1 0, gre vezje ob naslednjem urinem signalu v stanje S_2. V S_2 ostane, dokler je kombinacija signalov na vhodu 1 0 in nato še na 1 1. Pri dodatnem zasuku vretena je kombinacija vhodnih signalov 0 1. Ker je prišlo do polovičnega obrata vretena (v desno), se prvi izhod postavi na 1. Če se v stanju S_2 pojavi kombinacija vhodnih signalov 0 0, pomeni, da zasuka za pol obrata ni bilo in vezje se vrne v izhodiščno stanje S_0, izhoda pa sta oba na 0.



Slika 7.10-1. Senzorji na vretenu s pripadajočimi signali.

Če na vhodih ni sprememb, sta oba izhoda na 0. Ko pa je vhodna sekvenca takšna, da ustreza eni izmed smeri vrtenja, ustrezni izhod postane 1 za dolžino ene urine periode. V tabeli 7.10-1 so podane smeri vrtenja s pripadajočim stanjem in kodo stanja.



Slika 7.10-2: Diagram prehajanja stanj.

Tabela 7.10-1: Kodiranje stanj.

Smer vrtenja	Stanje	Koda stanj
Desno	S_2	1 0
Levo	S_1	0 1
Stoj	S_0	0 0

Pravilnostno tabelo za diagram prehajanja stanj zapišemo tako, da v prvi stolpec tabele 7.10-2 vpišemo vse kombinacije notranjih stanj, v naslednje stolpce pa naslednja stanja in izhode, ki so odvisni od trenutnega stanja in vhodov.

Tabela 7.10-2. Mealyjevo vezje lahko realiziramo z dvema pomnilnima celicama JK.

$Y_1\ Y_2$	$X_1\ X_2$			
	00	**01**	**10**	**11**
00	00/00	01/00	10/00	XX/XX
01	00/00	01/00	00/01	01/00
10	00/00	00/10	10/00	10/00
11	XX/XX	XX/XX	XX/XX	XX/XX

Tabela 7.10-3. Tabela za realizacijo vezja s pomnilnima celicama J K.

	$X_1\ X_2$	$Y_1\ Y_2$	$Y_1^+\ Y_2^+$	$Z_1\ Z_2$	$J_1\ K_1$	$J_2\ K_2$
0	0 0	0 0	0 0	0 0	0 X	0 X
1	0 0	0 1	0 0	0 0	0 X	X 1
2	0 0	1 0	0 0	0 0	X 1	0 X
3	0 0	1 1	X X	X X	X X	X X
4	0 1	0 0	0 1	0 0	0 X	1 X
5	0 1	0 1	0 1	0 0	0 X	X 0
6	0 1	1 0	0 0	1 0	X 1	0 X
7	0 1	1 1	X X	X X	X X	X X
8	1 0	0 0	1 0	0 0	1 X	0 X
9	1 0	0 1	0 0	0 1	0 X	X 1
10	1 0	1 0	1 0	0 0	X 0	0 X
11	1 0	1 1	X X	X X	X X	X X
12	1 1	0 0	X X	X X	X X	X X
13	1 1	0 1	0 1	0 0	0 X	X 0
14	1 1	1 0	1 0	0 0	X 0	0 X
15	1 1	1 1	X X	X X	X X	X X

Ker ne potrebujemo vseh možnih kombinacij, označimo odvečne z X. To upoštevamo pri minimizaciji s Karnaughovimi diagrami.

J_1:

$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	0	0	X	1
01	0	0	0	0
11	X	X	X	X
10	X	X	X	X

$$J_1 = X_1 \overline{Y_2}$$

J_2:

$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	0	1	X	0
01	X	X	X	X
11	X	X	X	X
10	0	0	0	0

$$J_2 = X_2 \overline{Y_1}$$

K_1:

$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	1	1	0	0

$$K_1 = \overline{X_1}$$

K_2:

$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	X	X	X	X
01	1	0	0	1
11	X	X	X	X
10	X	X	X	X

$$K_2 = \overline{X_2}$$

Z_1:

$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	0	0	X	0
01	0	0	0	0
11	X	X	X	X
10	0	1	0	0

$$Z_1 = \overline{X_1} X_2 Y_1$$

Z_2:

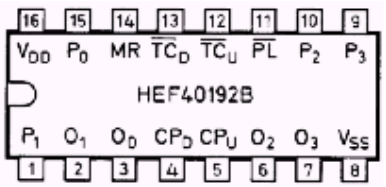
$Y_1 Y_2$ \ $X_1 X_2$	00	01	11	10
00	0	0	X	0
01	0	0	0	1
11	X	X	X	X
10	0	0	0	0

$$Z_2 = X_1 \overline{X_2} Y_2$$

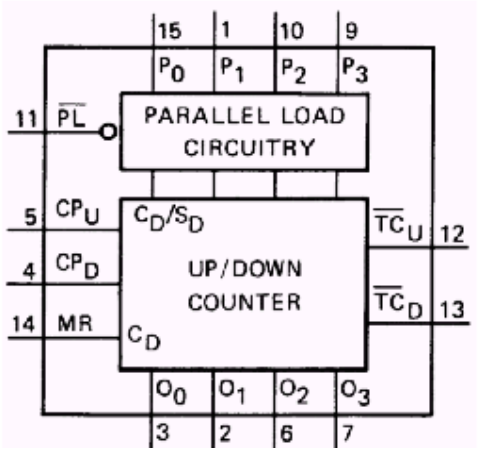
Števnik

Na izhoda inkrementalnega dajalnika priključimo števnik: ob vrtenju v desno bo števnik štel navzgor, ob vrtenju v levo pa navzdol. Izhod števnika peljemo na dekodirnik, ki prekodira kod BCD v kod, ki krmili 7-segmentni prikazovalnik. Več števnikov lahko povežemo tudi v zaporedje glede na število obratov, ki jih želimo prešteti.

Vezje ima dva ločena izhoda za *gor* in *dol*, zato izberemo števnik HEF40192. To je števnik, ki ima tudi možnost začetne nastavitve (nastavitev pozicije) in ločene vhode za štetje navzgor in navzdol. Priključna shema števnika HEF40192 je prikazana na sliki 7.10-3, njegova notranja zgradba pa na sliki 7.10-4.



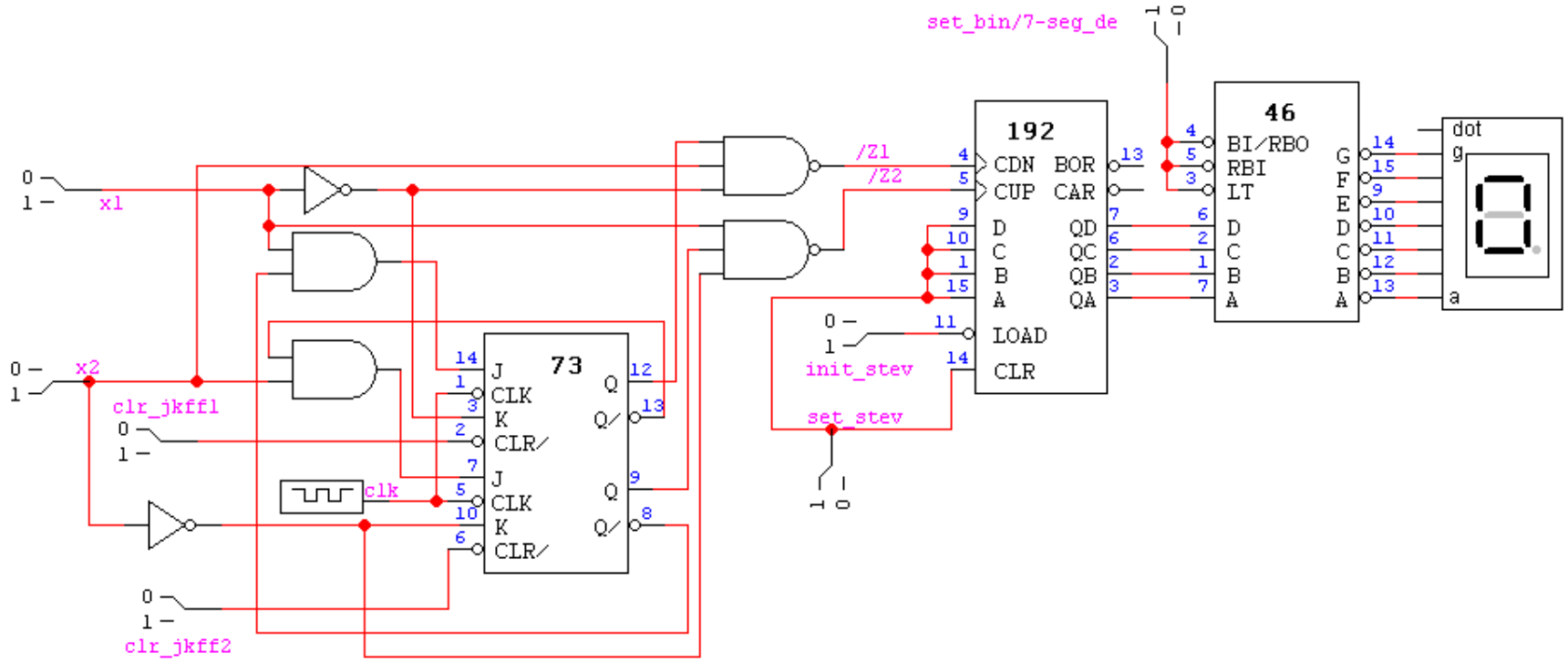
Slika 7.10-3. Priključna shema za števnik HEF40192.



Slika 7.10-4. Notranja zgradba števnika HEF40192.

Zaključek

Celotno vezje inkrementalnega dajalnika je na sliki 7.10-5.



Slika 7.10-5. Celotno vezje inkrementalnega dajalnika.

8 PRILOGE

8.1 Priloga A: Elementi

9 LITERATURA

Za kvalitetno in nemoteno izvajanje praktičnih vaj si mora študent pridobiti dovolj teoretičnega predznanja. Osnovno študijsko gradivo so zapiski predavanj in viri [1], [2], [3] in [4]. Za poglobljen študij teorije preklopnih vezij in končnih avtomatov navajamo tudi klasične vire [5], [6], [7] in [8]. Programski paket LogicWorks je opisan v [9] in [10] .

[1] Zmago Brezočnik, Preklopne strukture in sistemi, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, v pripravi.

[2] Randy H. Katz, Contemporary Logic Design, The Benjamin/Cummings Publishing Company, 1994.

[3] John P. Hayes, Digital Logic Design, Addison-Wesley, 1993.

[4] Jernej Virant, Logične osnove odločanja in pomnjenja v računalniških sistemih, Fakulteta za elektrotehniko in računalništvo, Ljubljana, 1990.

[5] Zvi Kohavi, Switching and Finite Automata Theory, McGraw-Hill, 1970.

[6] Samuel C. Lee, Modern Switching Theory and Digital Design, Prentice-Hall, 1978.

[7] J. Hartmanis, Algebraic Structure Theory of Sequential Machines, Prentice-Hall, 1966.

[8] A. Harrison, Introduction to Switching and Automata Theory, McGraw-Hill, 1965.

[9] Capilano Computing Systems, LogicWorks 3, Addison-Wesley Publishing Company, 1996.

[10] Janez Stergar, Bogomir Horvat, Digitalne strukture, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, 1997.