8 SINTEZA SINHRONIZIRANIH SEKVENČNIH VEZIJ

8. 1 Sinteza iz besednega opisa

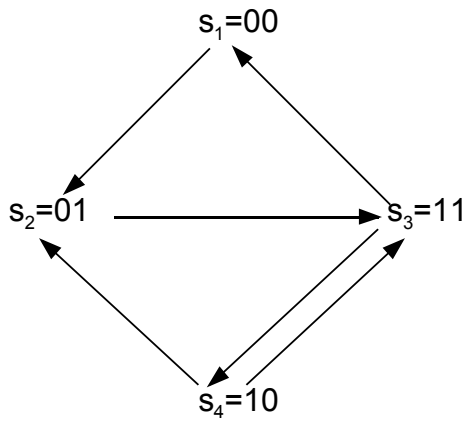
Zgled: Detektor sekvence.

Sinhronski sekvenčni avtomat z dvema vhodnima in dvema izhodnima stanjema mora dati na izhodu stanje 1 vsakič, ko je na vhodu detektirana sekvenca 0101, v vseh ostalih primerih je izhodno stanje 0.

To pomeni, da vhodni sekvenci 010101 pripada izhodna sekvenca v časovnem zaporedju 000101.
Zgradimo tabelo prehajanja stanj, ki s formalnim jezikom opisuje delovanje avtomata.

S	**Δ^1s, z**	
	x_1 = 0	**x_2 = 1**
s_1	**s_2, 0**	**s_1, 0**
s_2	**s_2, 0**	**s_3, 0**
s_3	**s_4, 0**	**s_1, 0**
s_4	**s_2, 0**	**s_3, 1**

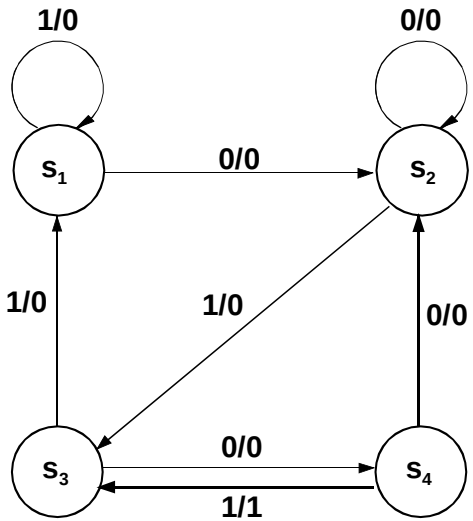
y_1y_2	$\Delta^1y_1\ \Delta^1y_2$		z	
	x = 0	x = 1	x = 0	x = 1
00	01	00	0	0
01	01	11	0	0
11	10	00	0	0
10	01	11	0	1



$$\Delta^1 y_1 = \bar{x}y_1y_2 + x\bar{y}_1y_2 + xy_1\bar{y}_2$$

$$\Delta^1 y_2 = y_1\bar{y}_2 + \bar{x}\bar{y}_1\bar{y}_2 + \bar{y}_1y_2$$

$$z = xy_1\bar{y}_2$$



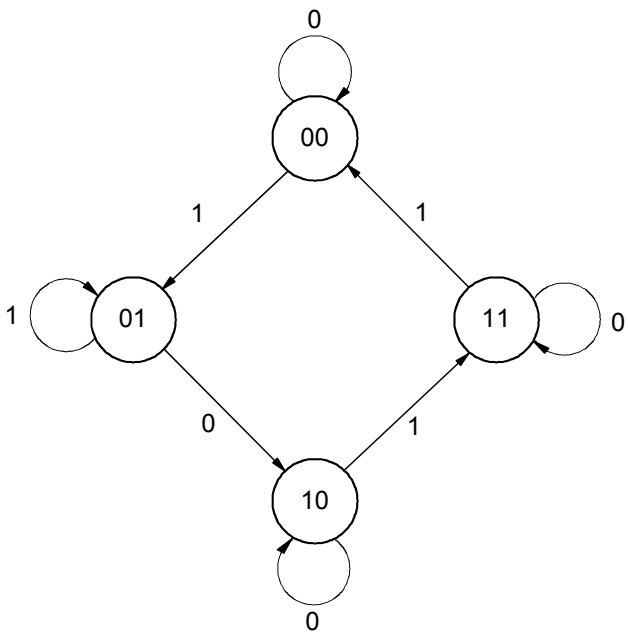
y_1y_2	$\Delta^1y_1\,\Delta^1y_2$		z	
	x = 0	x = 1	x = 0	x = 1
00	01	00	0	0
01	01	10	0	0
10	11	00	0	0
11	01	10	0	1

$\Delta^1y_1 = \bar{x}y_1\bar{y}_2 + xy_2$ $\quad\Delta^1y_2 = \bar{x}$ $\quad z = xy_1y_2$

8. 2 Sinteza vezja iz diagrama prehajanja stanj

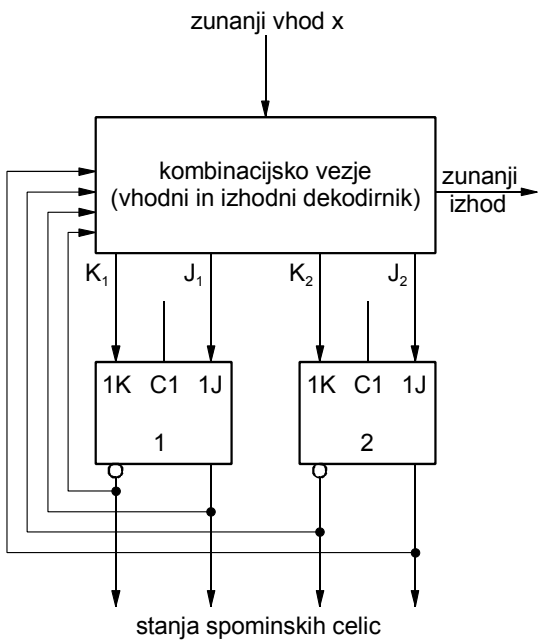
<u>Zgled:</u>

Konstruirati želimo sekvenčno vezje s spodnjim diagramom stanj. Odločili smo se za uporabo JK spominskih celic.



Sed. stanje		Naslednje stanje			
		Sed. x = 0		Sed. x = 1	
y_1	y_2	Δ^1y_1	Δ^1y_2	Δ^1y_1	Δ^1y_2
0	0	0	0	0	1
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	1	0	0

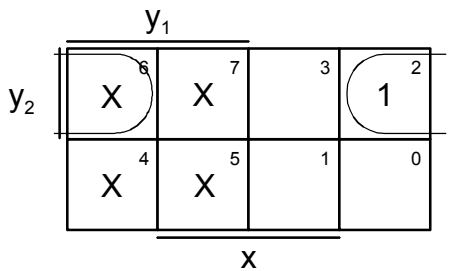
Vhodi v odločit. vezje			Naslednje		Izhodi iz odločitvenega vezja			
Sed. stanje		vhod	stanje		Vhodi v spominske celice			
y_1	y_2	x	$\Delta^1 y_1$	$\Delta^1 y_2$	J_1	K_1	J_2	K_2
0	**0**	**0**	**0**	**0**	**0**	**X**	**0**	**X**
0	**0**	**1**	**0**	**1**	**0**	**X**	**1**	**X**
0	**1**	**0**	**1**	**0**	**1**	**X**	**X**	**1**
0	**1**	**1**	**0**	**1**	**0**	**X**	**X**	**0**
1	**0**	**0**	**1**	**0**	**X**	**0**	**0**	**X**
1	**0**	**1**	**1**	**1**	**X**	**0**	**1**	**X**
1	**1**	**0**	**1**	**1**	**X**	**0**	**X**	**0**
1	**1**	**1**	**0**	**0**	**X**	**1**	**X**	**1**



Vhodne spremenljivke so y_1,y_2,x, izhodi pa K_1, J_1, K_2, J_2.

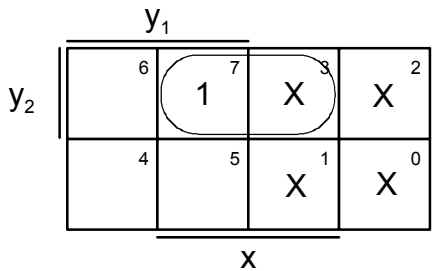
y_1	y_2	x	J_1	K_1	J_2	K_2
0	0	0	0	X	0	X
0	0	1	0	X	1	X
0	1	0	1	X	X	1
0	1	1	0	X	X	0
1	0	0	X	0	0	X
1	0	1	X	0	1	X
1	1	0	X	0	X	0
1	1	1	X	1	X	1

$J_1 = \bar{y}_1 y_2 \bar{x} +$ štiri neopredelejene kombinacije



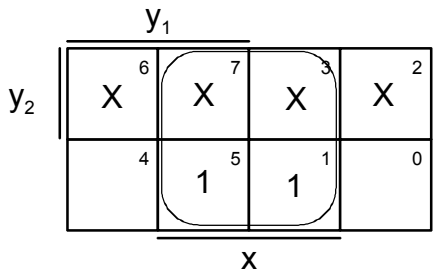
$J_1 = y_2 \bar{x}$

$K_A = y_1 y_2 x +$ štiri neopredeljene kombinacije



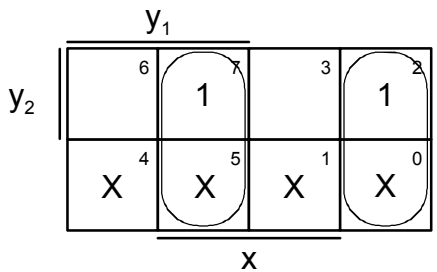
$K_1 = y_2 x$

$J_2 = \bar{y}_1 \bar{y}_2 x + y_1 \bar{y}_2 x +$ štiri neopredeljene kombinacije

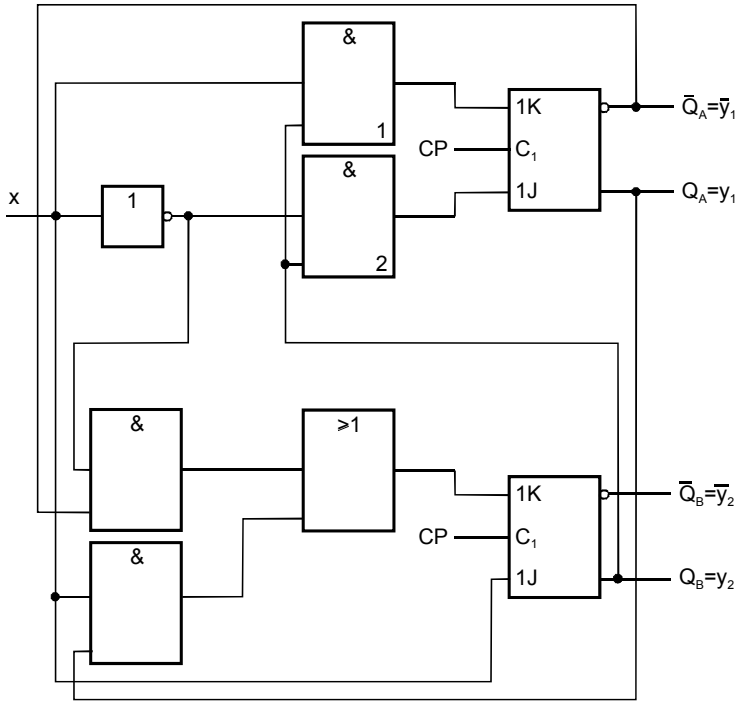


$J_2 = x$

$K_2 = \bar{y}_1 y_2 \bar{x} + y_1 y_2 x +$ štiri neopredeljene kombinacije



$K_2 = y_1 x + \bar{y}_1 \bar{x}$



8. 3. Sinteza vezja iz časovnega diagrama

Konstruirajte vezje z spominsko celico Q in dvema vhodoma x_1 in x_2 tako, da bo realiziran spodnji časovni diagram.

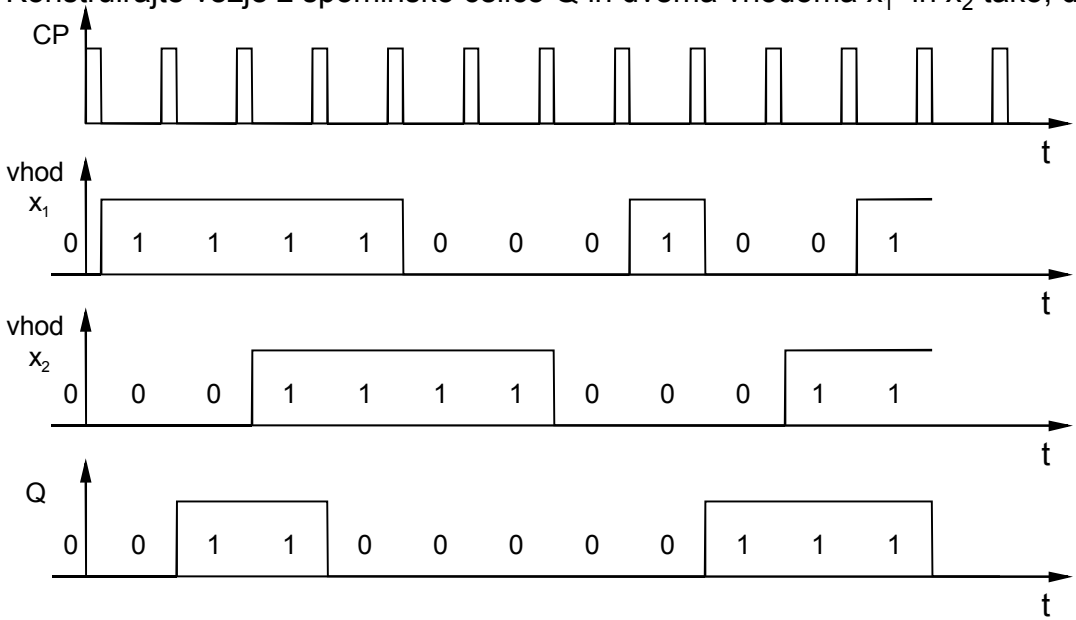


Diagram torej zahteva, da postane Q = 1, če je x_1 = 1 in x_2 = 0 ter Q = 0, če je x_1 = 1 in x_2 = 1 pri ostalih vhodnih kombinacijah x_1 = 0, x_2 = 0 ter x_1 = 0, x_2 = 1 pa naj ostane stanje nespremenjeno.

Funkcija S je:

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

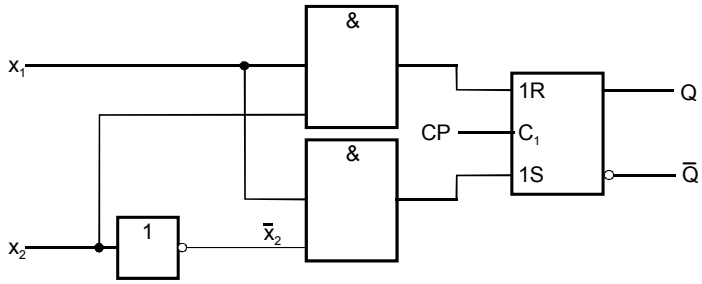
$S = x_1\bar{x}_2$

Funkcija R pa:

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

$R = x_1x_2$

Sed. stanje	Sedanja vhoda		Nasled. stanje	Vhodi k spom. celicam	
Q	x_1	x_2	Δ^1Q	S	R
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	0
0	1	1	0	0	X
1	0	0	1	X	0
1	0	1	1	X	0
1	1	0	1	X	0
1	1	1	0	0	1

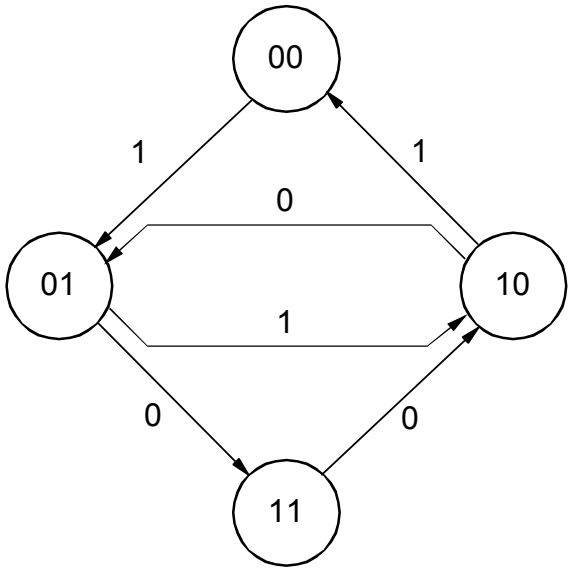


Sinteza nepopolno opredeljenega avtomata

Zgled:

Sekvenčno vezje z dvema spominskima celicama in enim vhodom mora dajati naslednje sekvence. Če je vhod = 1, naj dajeta celici na svojih izhodih sekvenco 00 01 10, če pa je vhod = 0 naj se ponavlja sekvenca 11, 10, 01.

Avtomat mora imeti naslednji diagram prehajanja stanj:



Določitev neopredeljenih stanj je prepuščena konstruktorju, ki jih lahko izbere na več načinov. Lahko pa jih pusti, kot neopredeljena; torej poljubna in jih v postopku sinteze izkoristi za poenostavitev vhodnega dekodirnika.

Reši to nalogo kot popolno opredeljen in kot nepopolno opredeljen avtomat in primerjaj dobljeni rezultat.

8. 4 Števci

Števci tvorijo zelo pomembno in raznoliko skupino sekvenčnih vezij, ki so kot podsestavi nepogrešljiv del skoraj vsake obsežnejše digitalne naprave. Za razliko od običajnih sekvenčnih vezij velikokrat sploh nimajo formalnega vhoda. Vzbujajo jih števni impulzi, ki so lahko impulzi ure, izhodni impulzi drugih vezij, kot so na primer dajalniki pozicije in podobno.

Doseg štetja števcev je določen s številom možnih stanj, v katerih se ta lahko nahaja preden se vrne v začetno stanje, ki je najpogosteje stanje, ki predstavlja ničlo.

0, 1, 2, ... , m, 0, 1, 2,...
$m \leq 2^n - 1$

Prednastavitev in brisanje: Set, Reset – n+1 vhodov pri n izhodih.

Delilniki frekvence: 1 vhod in 1izhod – izhod se spremeni vsakih d vhodnih impulzov

Števec, ki šteje 0, 1, 2,..., m, je mogoče zgraditi z delilnikom z (m+1)

Delilnik frekvence z "d" je mogoče zgraditi s poljubnim števcem, ki šteje vsaj od 0, 1, 2,.., d -1, 0, 1, 2,...

8. 4. 1. Razdelitev števcev

- navadni ali enostavni števci (0, 1, 2,...., m, 0, 1, 2,.....) $m \leq 2^n - 1$
- posebni ali specialni števci

Vrste navadnih števcev:

- binarni števci z različnimi kodami
- dekadni števci
- števci po modulu m
- delilniki s številom "d"

Posebni ali specialni števci:

- dvosmerni števci
- krožni števci
- števci s prepletanjem
- števno dekodirna vezja

8. 4. 2 Binarni števci

Binarni števci so števci, ki štejejo preko vseh možnih stanj; to je 2^n. Njihov izhod tako predstavlja binarno kodirane števke desetiškega številskega sistema ali pa dekadna števila, če je število spominskih celic enako ali večje od štiri. Za sintezo takšnega števca zadošča že diagram zaporednih prehodov stanj, ki je v tem primeru določen s binarnim številskim sistemom.

8. 4. 2. 1 Navadni štiribitni binarni števec:

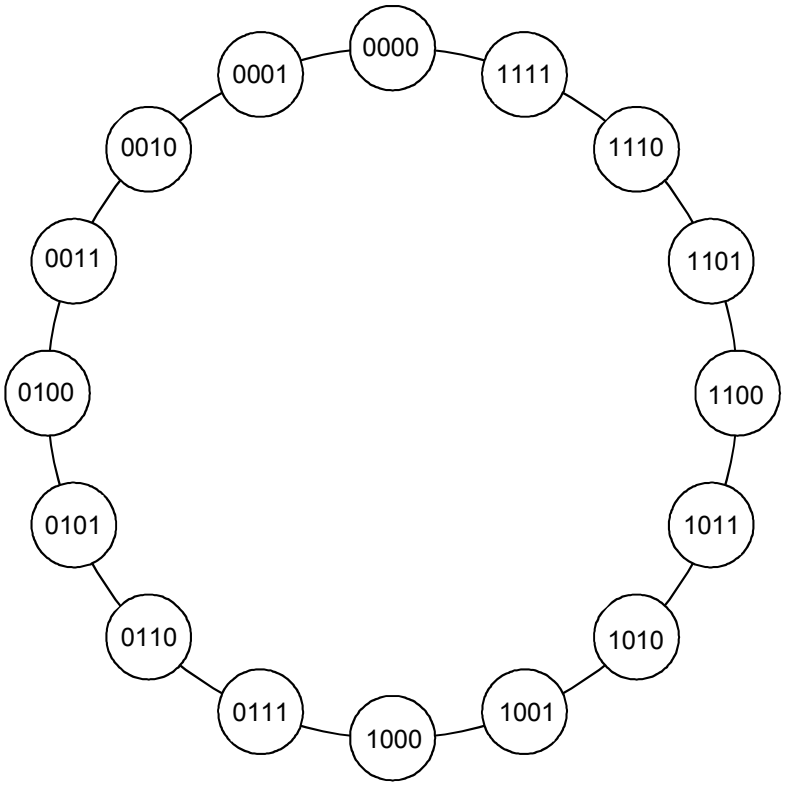
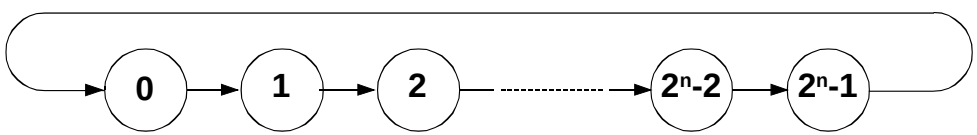
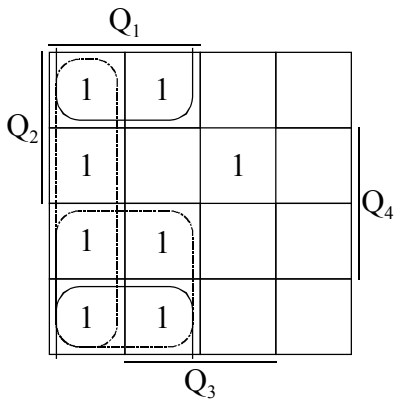


Diagram prehajanja stanj tega števca pa lahko podamo tudi z dekdnimi števkami oziroma številom, saj je pravilo kodiranja stanj povsem določeno.

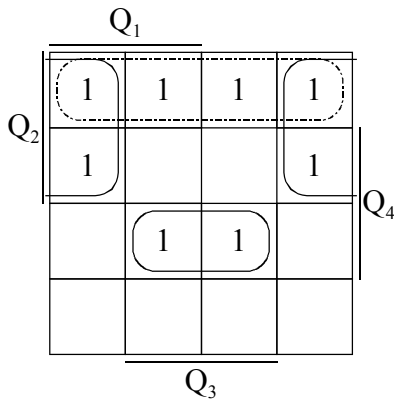


Q_1	Q_2	Q_3	Q_4	D_1	D_2	D_3	D_4
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

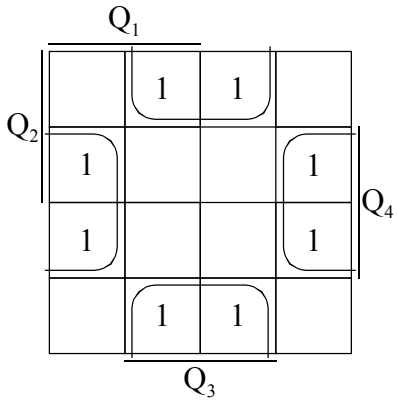
D_1:



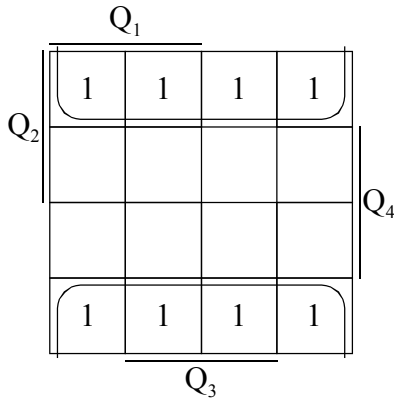
D_2:



D_3:



D_4:

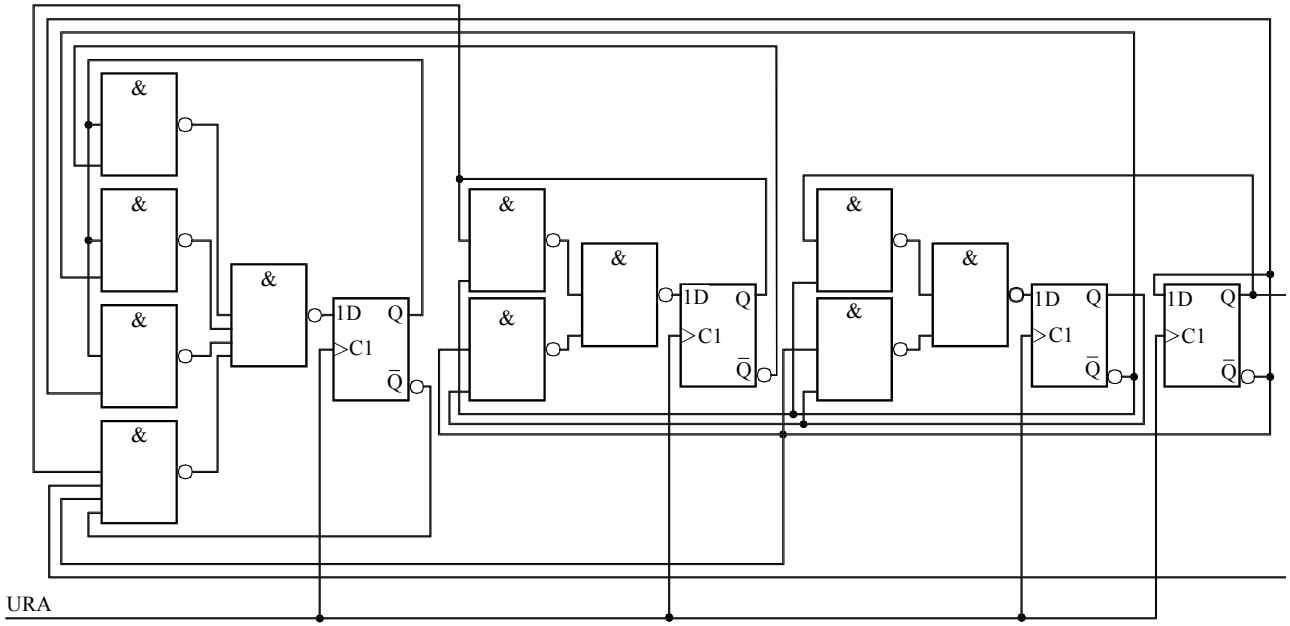


$$D_1 = \overline{Q}_1 Q_2 Q_3 Q_4 + Q_1 \overline{Q}_3 + Q_1 \overline{Q}_2 + Q_1 \overline{Q}_4$$

$$D_2 = Q_2 \overline{Q}_4 + Q_2 \overline{Q}_3 + \overline{Q}_2 Q_3 Q_4$$

$$D_3 = Q_3 \overline{Q}_4 + \overline{Q}_3 Q_4$$

$$D_4 = \overline{Q}_4$$



8.4. 2. 2 Enostaven binarni števec z vključenim prenosom

$$Q_1 = Q_3, \quad Q_2 = Q_2, \quad Q_3 = Q_1, \quad Q_4 = Q_0$$

$$D_0 = \overline{Q}_0$$

$$D_1 = Q_1 \oplus Q_0$$

$$D_2 = Q_2 \oplus Q_1 Q_0$$

$$D_3 = Q_3 \oplus Q_2 Q_1 Q_0$$

$$D_i = Q_i \oplus Q_{i-1} Q_{i-2} \ldots Q_1 Q_0 = Q_i \oplus \left[\mathop{\&}_{j=0}^{i-1} Q_j \right] \quad \text{pri čemer je } 1 \le i \le n$$

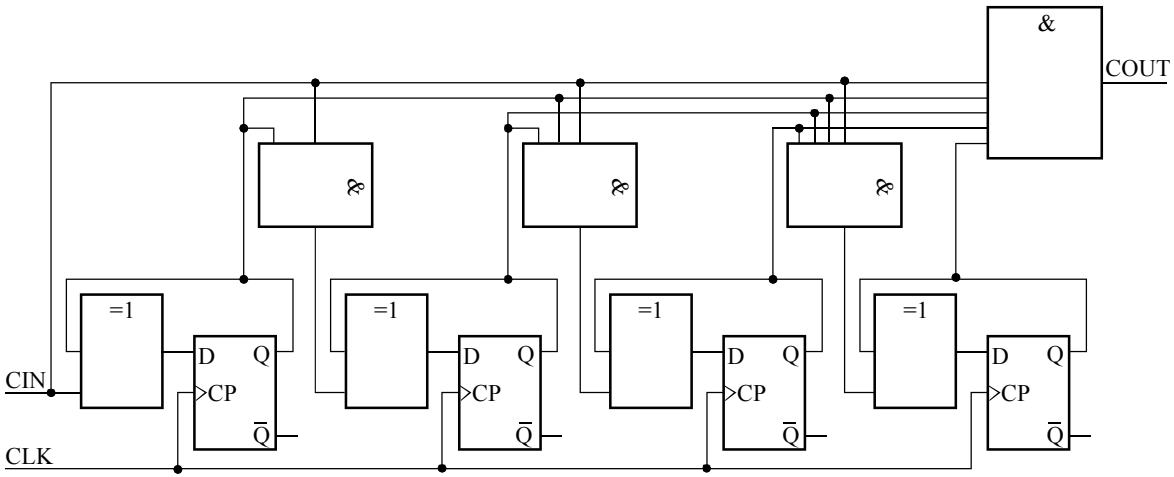
$$D_0 = \overline{Q}_0 = Q_0 \oplus 1$$

$$\Delta^1 Q_0 = \Delta^1 \overline{Q}_0 = Q_0 \oplus 1$$

$$\Delta^1 Q_1 = Q_1 \oplus Q_0$$

$$\Delta^1 Q_2 = Q_2 \oplus Q_1 Q_0$$

$$\Delta^1 Q_i = Q_i \oplus \left[\mathop{\&}_{j=0}^{i-1} Q_j \right] \quad \text{pri čemer je } 1 \le i \le n$$



8. 4. 3 Dekadni števci

Tudi dekadne števce lahko gradimo z različnimi kodami, glede na namen njihove uporabe. Tu pogosto srečamo prav naravno BCD kodo, kar pravzaprav predstavlja binarni števec s predčasnim vračanjem v izhodiščni položaj (0). Med temi števci je zelo znan tako imenovani Johnsonov dekadni števec, ki si ga bomo ogledali za navadnim dekadnim števcem. Temu števcu lahko rečemo tudi števec z naravno BCD kodo

8. 4. 3. 1 Dekadni števec v naravni BCD kodi

	Števno zaporedje				Vhodi spominskih celic			
n	Q_8	Q_4	Q_2	Q_1	T_{Q8}	T_{Q4}	T_{Q2}	T_{Q1}
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1
2	0	0	1	0	0	0	0	1
3	0	0	1	1	0	1	1	1
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	0	1	1
6	0	1	1	0	0	0	0	1
7	0	1	1	1	1	1	1	1
8	1	0	0	0	0	0	0	1
9	1	0	0	1	1	0	0	1
	0	0	0	0				

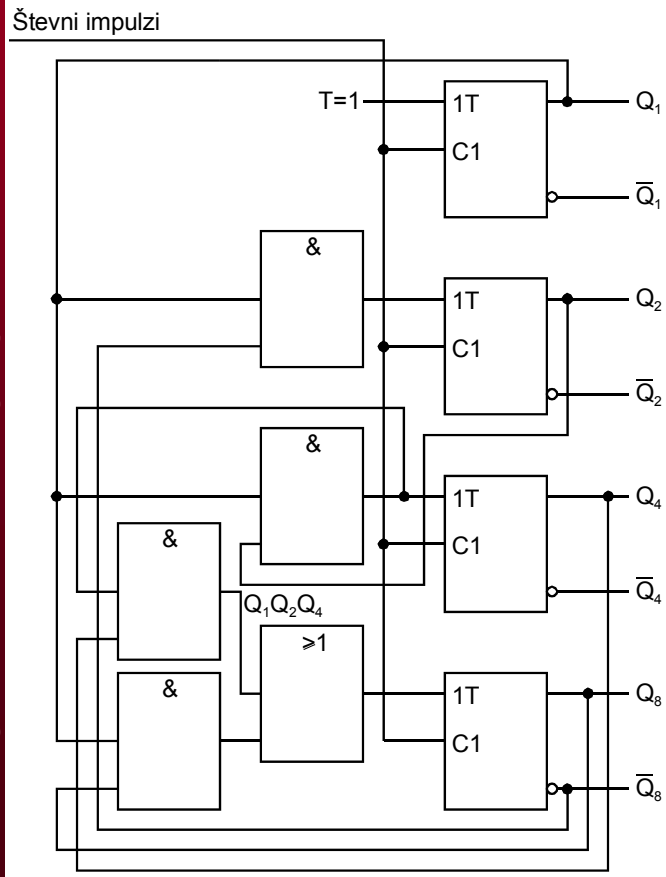
Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10			X	X

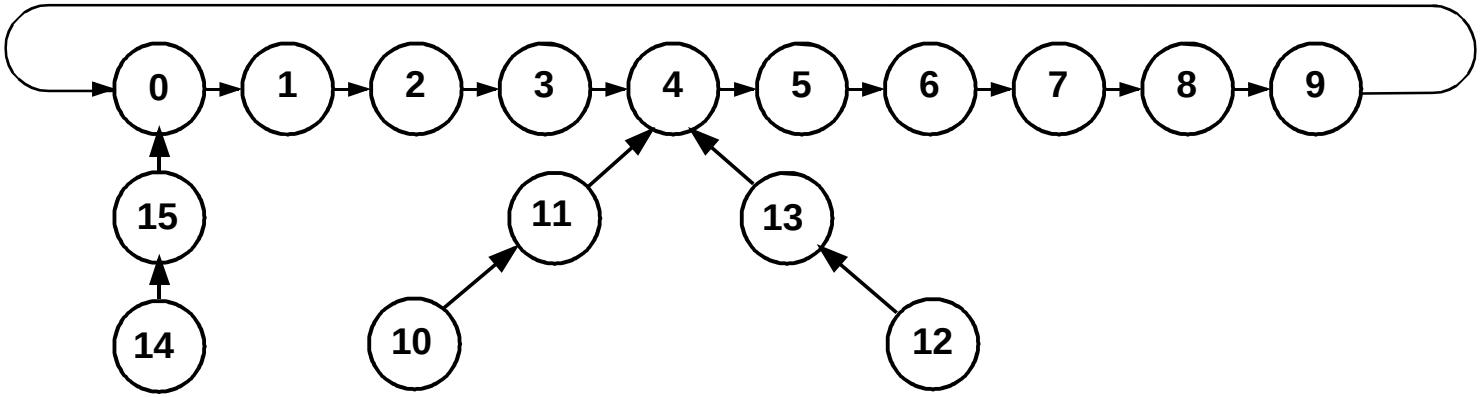
Q_8Q_4 \ Q_2Q_1	00	01	11	10
00			1	
01			1	
11	X	X	X	X
10			X	X

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00				
01			1	
11	X	X	X	X
10		1	X	X

$$T_{Q_1} = 1, \quad T_{Q_2} = \overline{Q}_8 Q_1, \quad T_{Q_4} = Q_2 Q_1, \quad T_{Q_8} = Q_8 Q_1 + Q_1 Q_2 Q_4$$



8. 4. 3. 2 Dekadni Johnsonov števec



N	Q_1	Q_2	Q_3	Q_4	Δ^1Q_1	Δ^1Q_2	Δ^1Q_3	Δ^1Q_4	J_1	K_1	J_2	K_2	J_3	K_3	J_4	K_4
0	0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
1	0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
2	0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
3	0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
4	0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
5	0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
6	0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
7	0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
8	1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
9	1	0	0	1	0	0	0	0	x	1	0	x	0	x	x	1
10	1	0	1	0	1	0	1	1	x	0	0	x	x	0	1	x
11	1	0	1	1	0	1	0	0	x	1	1	x	x	1	x	1
12	1	1	0	0	1	1	0	1	x	0	x	0	0	x	1	x
13	1	1	0	1	0	1	0	0	x	1	x	0	0	x	x	1
14	1	1	1	0	1	1	1	1	x	0	x	0	x	0	1	x
15	1	1	1	1	0	0	0	0	x	1	x	1	x	1	x	1

J_1:

$Q_2 \backslash Q_1$			
x	x		
x	x	1	
x	x		
x	x		

Q_3, Q_4

$$J_1 = Q_2 Q_3 Q_4$$

K_1:

$Q_2 \backslash Q_1$			
		x	x
1	1	x	x
1	1	x	x
		x	x

Q_3, Q_4

$$K_1 = Q_4$$

J_2:

$Q_2 \backslash Q_1$			
x	x	x	x
x	x	x	x
	1	1	

Q_3, Q_4

$$J_2 = K_2 = Q_3 Q_4$$

K_2:

$Q_2 \backslash Q_1$			
	1	1	
x	x	x	x
x	x	x	x

Q_3, Q_4

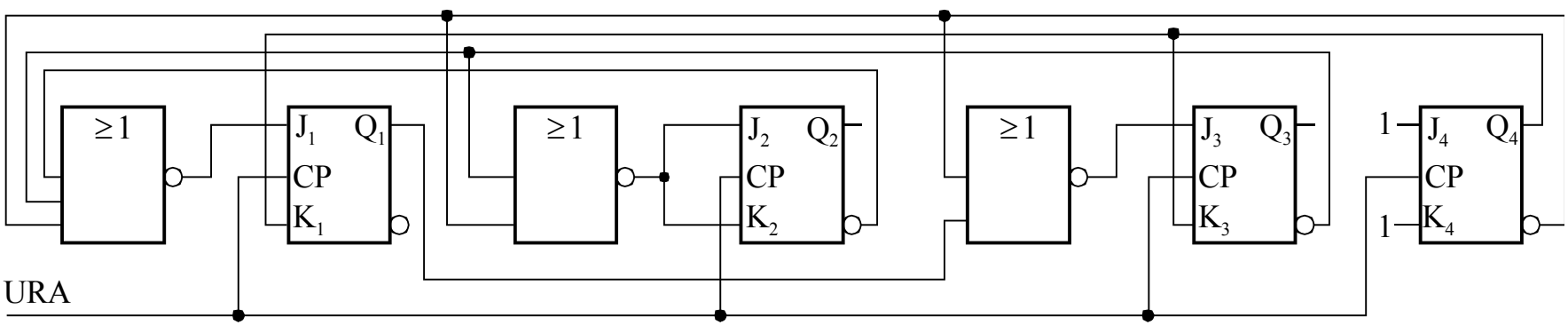
J_3:

	x	x	
	x	x	1
	x	x	1
	x	x	

K_3:

x			x
x	1	1	x
x	1	1	x
x			x

$$J_3 = \overline{Q}_1 Q_4;\ K_3 = Q_4$$

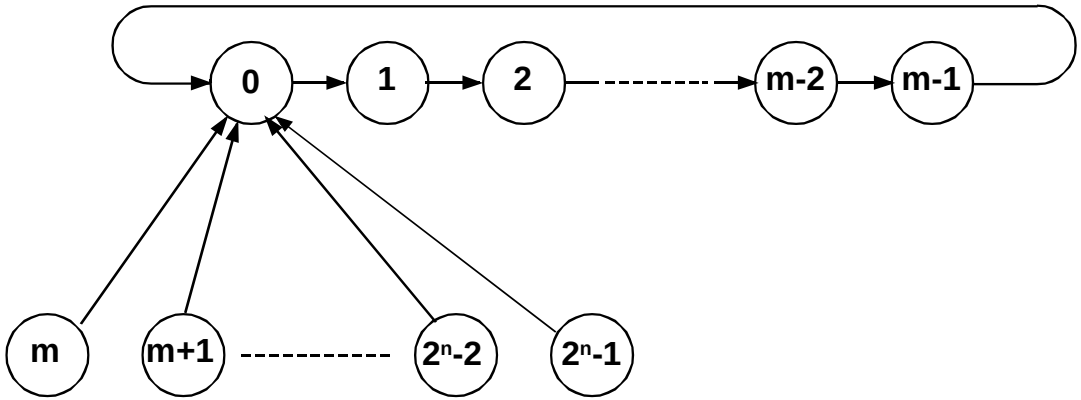


8. 4. 4 Števci po modulu "m"

Števci po modulu “m” štejejo od 0 do m-1.

0, 1, 2, ..., m - 1, 0, 1, 2, ... ; $m \leq 2^n - 1$

Imajo pa specifičen diagram prehajanja stanj, ki omogoča zanesljiv vstop v osnovni števni cikel.



8. 4. 4. 1 Števec po modulu 7

Sedanje stanje			Naslednje stanje			Vzbujalne funkcije					
Q_2	Q_1	Q_0	Δ^1Q_2	Δ^1Q_1	Δ^1Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	0	0	0	x	1	x	1	0	x
1	1	1	0	0	0	x	1	x	1	x	1

J_2:

Q_1 \ $\overline{Q_2}$			
	x	x	1
	x	x	

Q_0

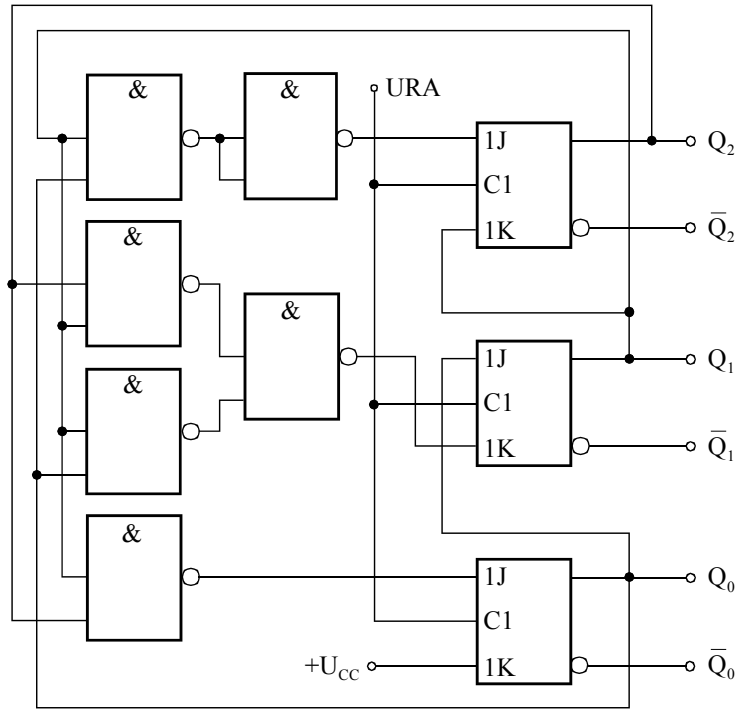
$J_2 = Q_1Q_0$

K_2:

Q_1 \ $\overline{Q_2}$			
	1	1	x	x
		x	x

Q_0

$K_2 = Q_1$



J_1:

Q_1 \ $\overline{Q_2}$			
	x	x	x	x
	x	1	1	

Q_0

$J_1 = Q_0$

K_1:

Q_1 \ $\overline{Q_2}$			
	1	1	1	
	x	x	x	x

Q_0

$K_1 = Q_1Q_2 + Q_1Q_0$

J_0:

Q_1 \ $\overline{Q_2}$			
		x	x	1
	1	x	x	1

Q_0

$J_0 = \overline{Q}_1 + \overline{Q}_2$

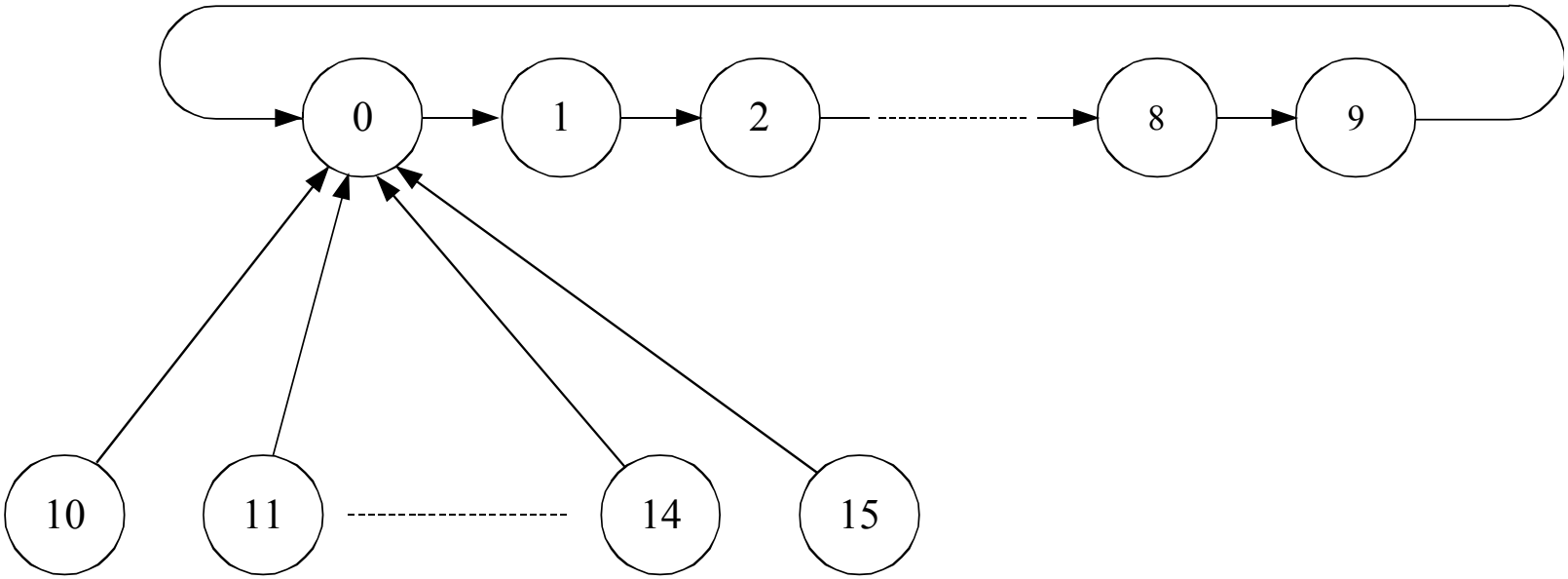
K_0:

Q_1 \ $\overline{Q_2}$			
	x	1	1	x
	x	1	1	x

Q_0

$K_0 = 1$

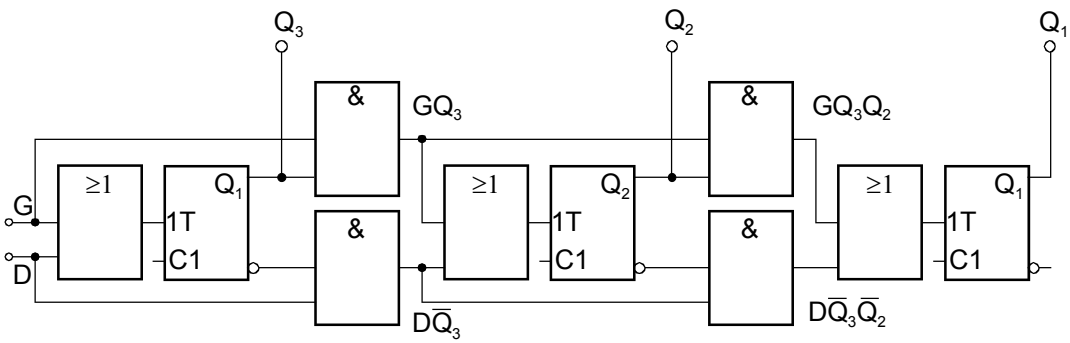
8. 4. 4. 2 Dekadni števec po modulu 10



8. 4. 5. Dvosmerni števci – dodamo en ali dva vhoda

G	Q_1	Q_2	Q_3	T_1	T_2	T_3
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	0	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	1
1	1	0	1	0	1	1
1	1	1	0	0	0	1
1	1	1	1	1	1	1

D	Q_1	Q_2	Q_3	T_1	T_2	T_3
1	0	0	0	1	1	1
1	1	1	1	0	0	1
1	1	1	0	0	1	1
1	1	0	1	0	0	1
1	1	0	0	1	1	1
1	0	1	1	0	0	1
1	0	1	0	0	1	1
1	0	0	1	0	0	1



8. 4. 5 Krožni števci

8. 4. 5. 1 Standardni krožni števci

Standardni krožni števec potrebuje “m” spominskih celic za štetje po modulu “m+1”

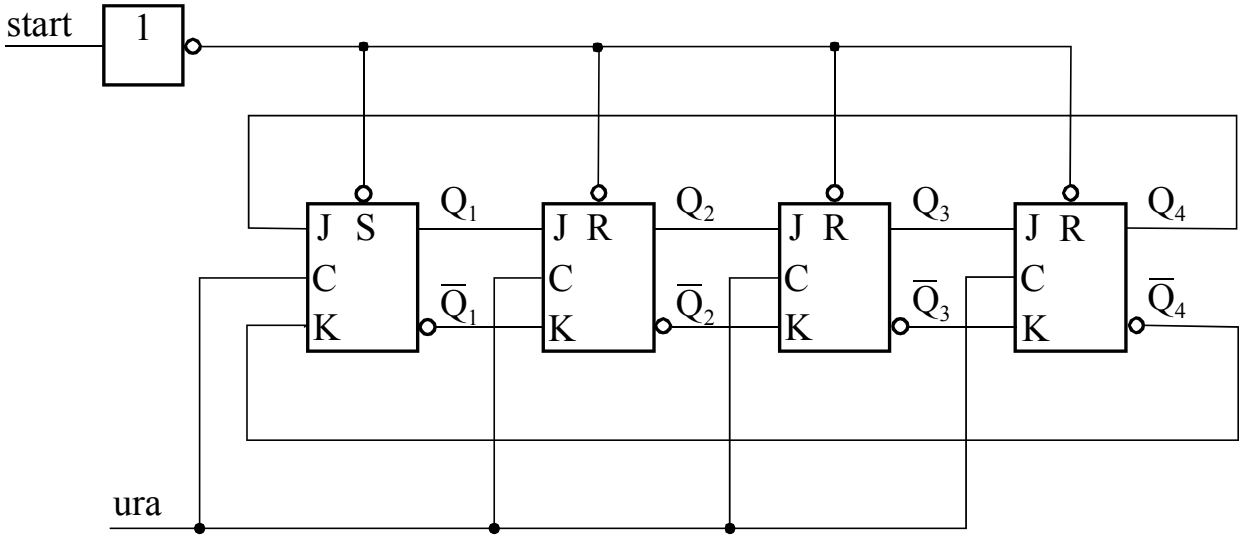
Primer standardnega krožnega števca:

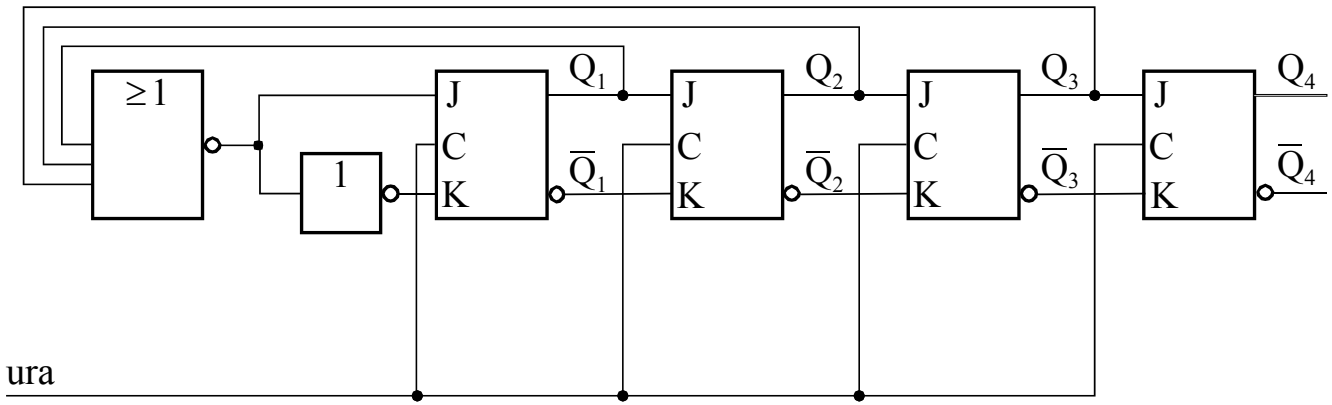
Q_1	Q_2	Q_3	Q_4
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

ali

Q_1	Q_2	Q_3	Q_4
0	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	0

Q_4	Q_3	Q_2	Q_1	Izhod
0	0	0	1	t_1
0	0	1	0	t_2
0	1	0	0	t_3
1	0	0	0	t_4


start
1
J S
C
K
J R
C
K
J R
C
K
J R
C
K
Q_1
$\overline{Q}_1$
Q_2
$\overline{Q}_2$
Q_3
$\overline{Q}_3$
Q_4
$\overline{Q}_4$
ura


≥1
1
J
C
K
Q_1
$\overline{Q}_1$
Q_2
$\overline{Q}_2$
Q_3
$\overline{Q}_3$
Q_4
$\overline{Q}_4$
ura

8. 4. 5. 2 Krožni števci s prepletanjem

Krožni števec s prepletanjem pa potrebuje “m/2” spominskih celic za štetje po modulu “m”

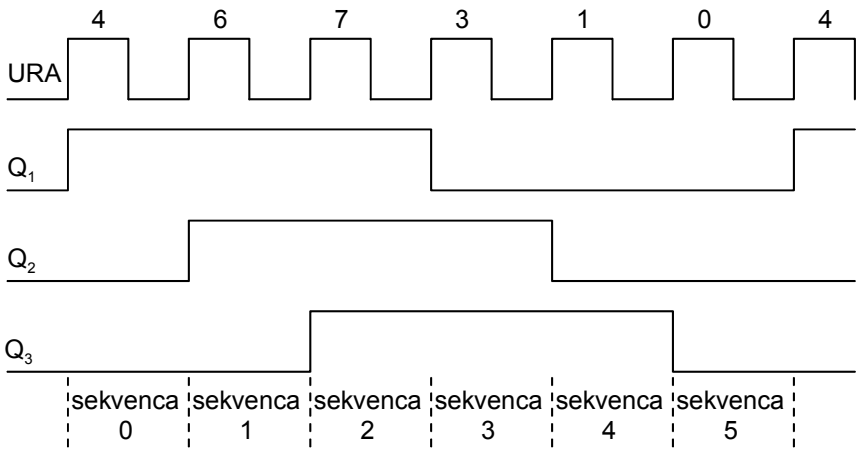
Primer števca s prepletanjem

Zanka nastane z invertiranjem enega od krajnih bitov (LSB ali MSB) in premikom na drugi konec

Q_1	Q_2	Q_3
0	0	0
1	0	0
1	1	0
1	1	1
0	1	1
0	0	1
0	0	0

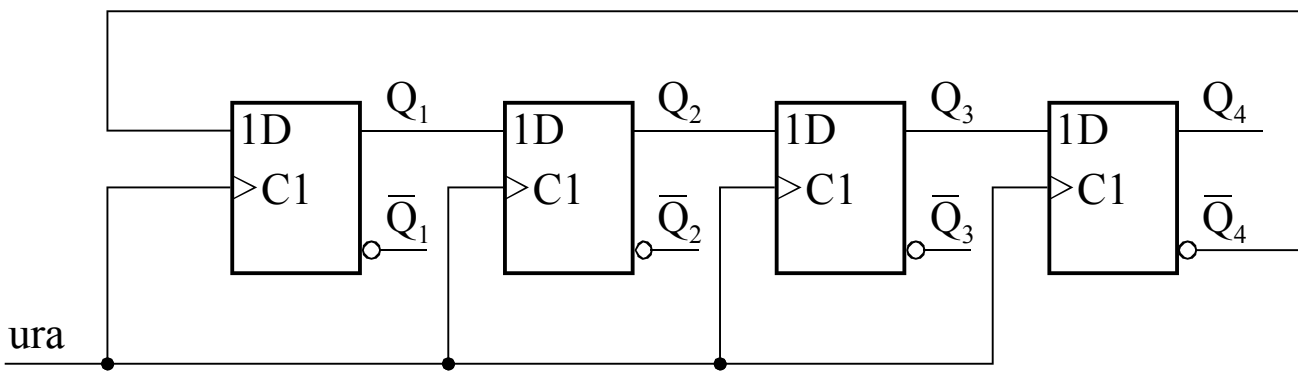
Q_1	Q_2	Q_3
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

Potrebujemo torej m/2 spominskih celic za štetje po modulu m



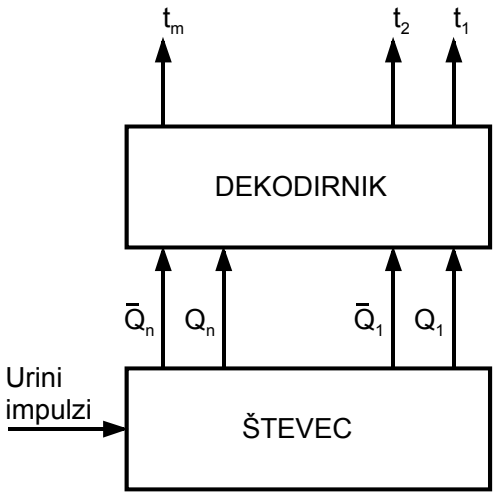
Če uporabimo 3/8 dekodirnik lahko dobimo šest ločenih časovnih impulzov.
Za to vrsto vezij uporabljamo skupno ime “števno dekodirna vezja”.

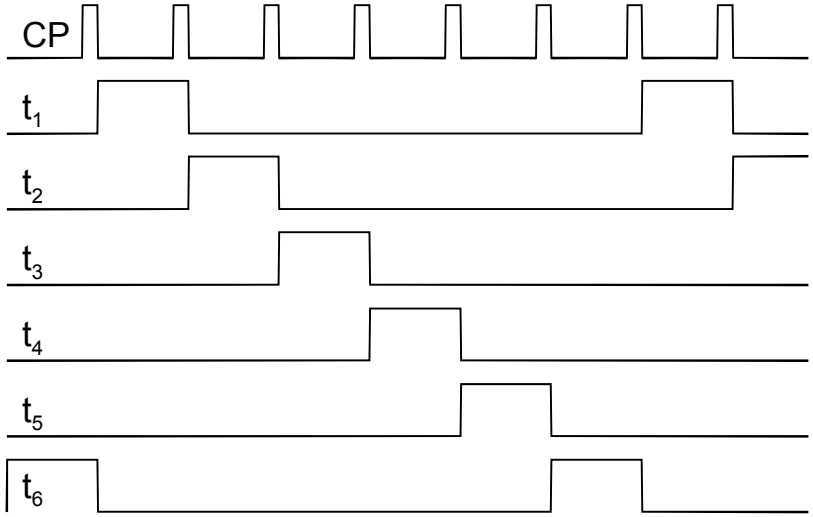
Indeks sekvence	Izhodi spominski celic				Izhodna funkcija
	Q_1	Q_2	Q_3	Q_4	
1	0	0	0	0	$\overline{Q}_1\overline{Q}_4$
2	1	0	0	0	$Q_1\overline{Q}_2$
3	1	1	0	0	$Q_2\overline{Q}_3$
4	1	1	1	0	$Q_3\overline{Q}_4$
5	1	1	1	1	Q_1Q_4
6	0	1	1	1	$\overline{Q}_1Q_2$
7	0	0	1	1	$\overline{Q}_2Q_3$
8	0	0	0	1	$\overline{Q}_3Q_4$



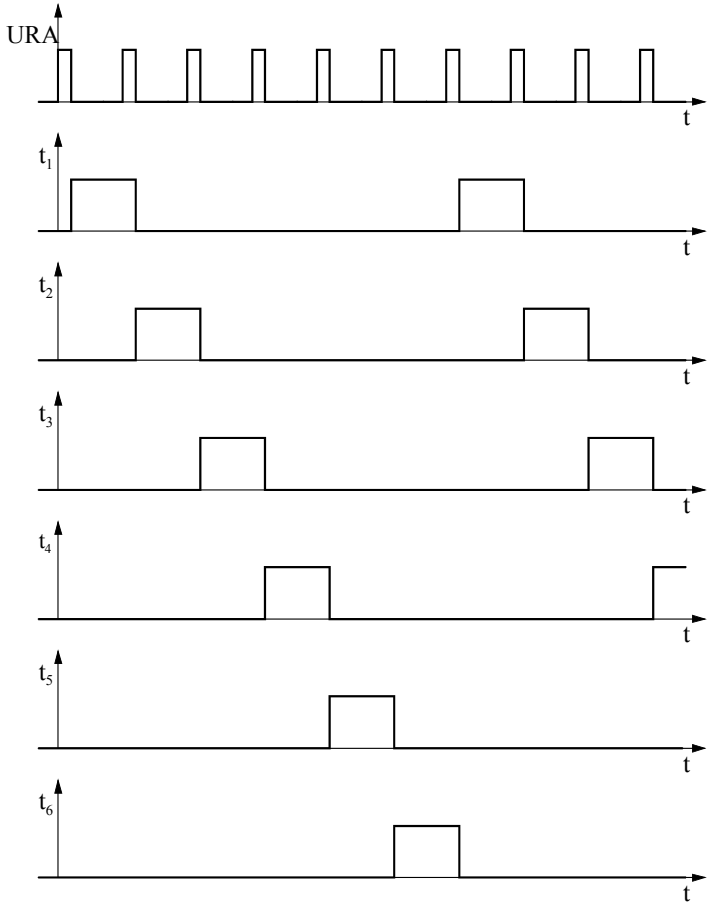
$$D_3 = (Q_1 + Q_3)Q_2$$

8. 4. 6 Števno dekodirna vezja

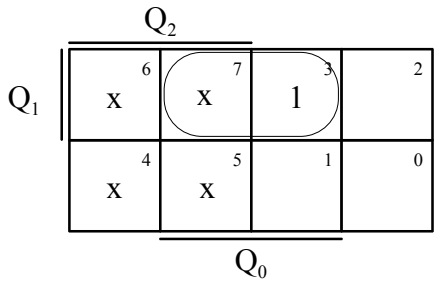




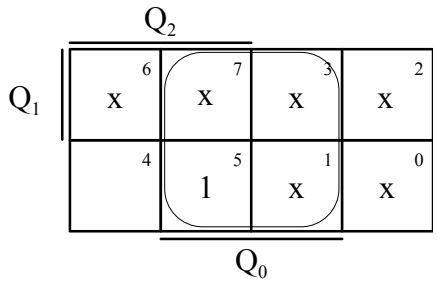
8. 4. 6. 1 Števno dekodirno vezje z dekodirnikom 3/6



J_2:



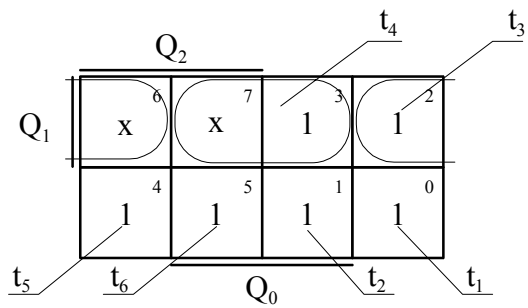
K_2:



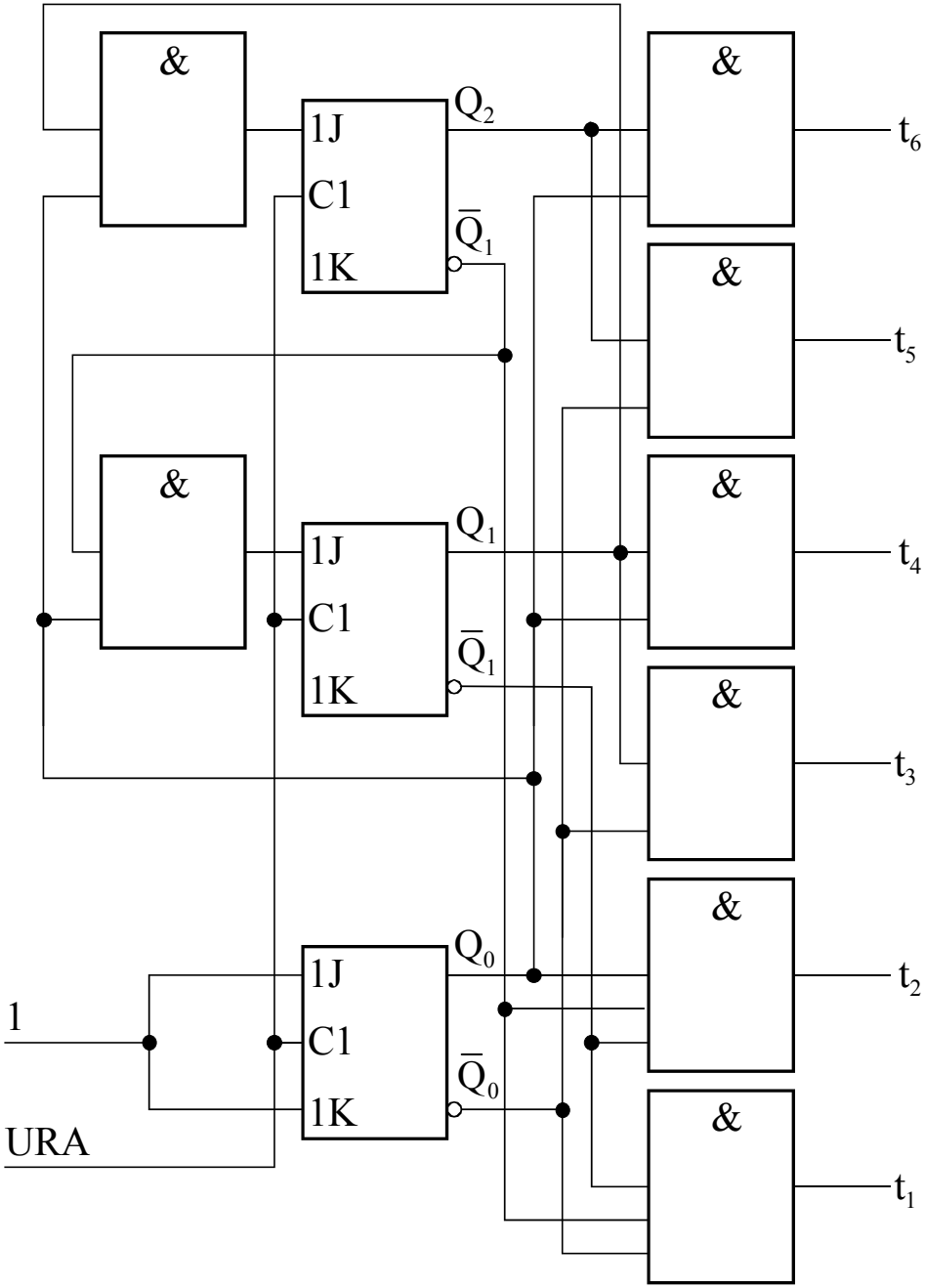
$J_2 = Q_1Q_0$; $K_2 = Q_0$

$J_1 = \overline{Q}_2Q_0$; $K_1 = Q_0$; $J_0 = 1$; $K_1 = 1$

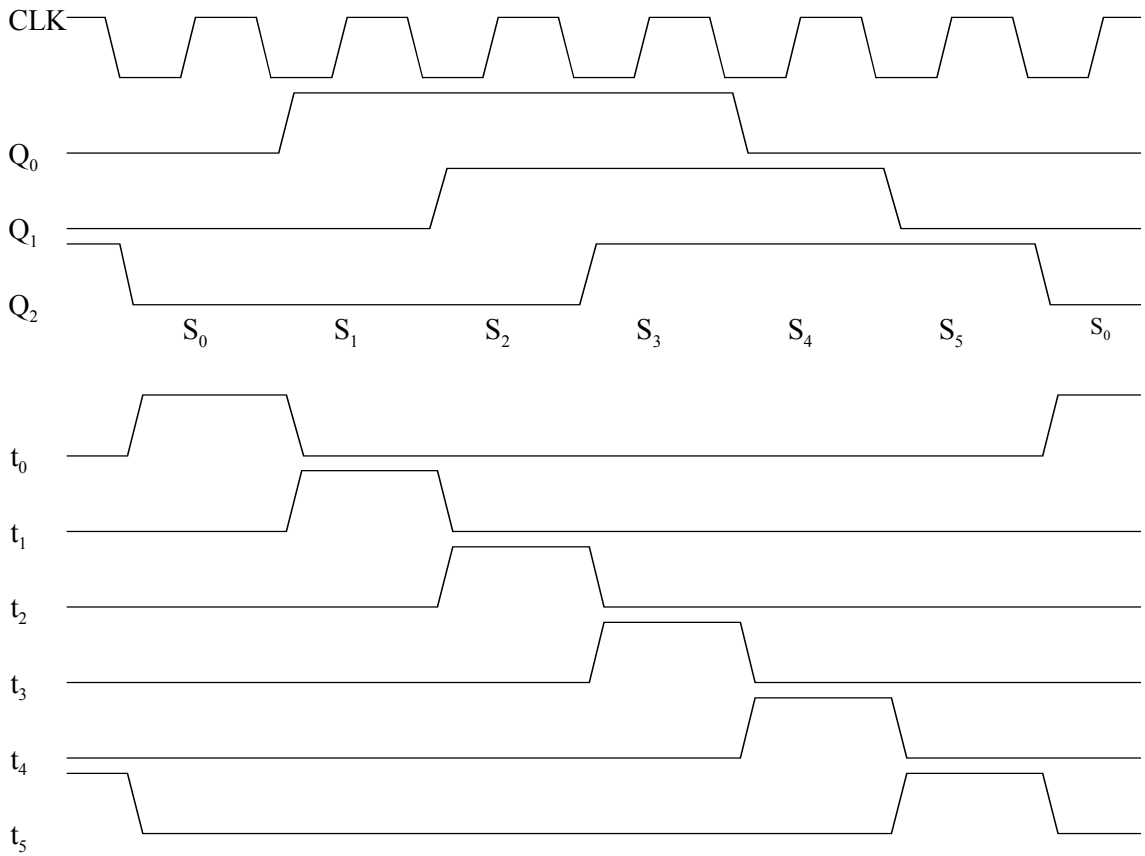
Št.:	Števno zaporedje			Vhodi spominskih celic					
	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	x	0	x	1	x
2	0	0	1	0	x	1	x	x	1
3	0	1	0	0	x	x	0	1	x
4	0	1	1	1	x	x	1	x	1
5	1	0	0	x	0	0	x	1	x
6	1	0	1	x	1	0	x	x	1
1	0	0	0	0	x	0	x	1	x

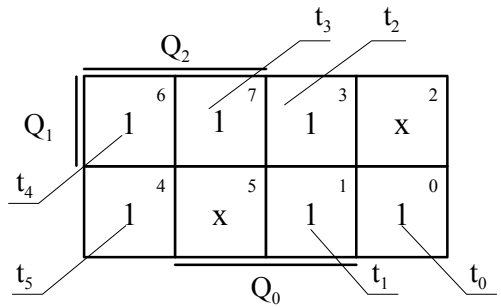


$t_1 = \overline{Q}_A\overline{Q}_2\overline{Q}_0$, $t_2 = \overline{Q}_2\overline{Q}_1Q_0$, $t_3 = Q_1\overline{Q}_0$, $t_4 = Q_1Q_0$, $t_5 = Q_2\overline{Q}_0$, $t_6 = Q_2Q_0$


&
1J
C1
1K
Q_2
$\overline{Q}_1$
Q_1
Q_0
$\overline{Q}_0$
1
URA
t_6
t_5
t_4
t_3
t_2
t_1

8. 4. 6. 2 Števno dekodirno vezje zasnovano s števcem s prepletanjem





$$S_0 = \overline{Q}_2\overline{Q}_1\overline{Q}_0; \quad S_1 = \overline{Q}_2\overline{Q}_1Q_0; \quad S_2 = \overline{Q}_2Q_1Q_0;$$

$$S_3 = Q_2Q_1Q_0; \quad S_4 = Q_2Q_1\overline{Q}_0; \quad S_5 = Q_2\overline{Q}_1\overline{Q}_0;$$

$$t_0 = \overline{Q}_0\overline{Q}_2; \quad t_1 = Q_0\overline{Q}_1; \quad t_2 = Q_0\overline{Q}_2; \quad t_3 = Q_0Q_2; \quad t_4 = \overline{Q}_0Q_1; \quad t_5 = \overline{Q}_1Q_2;$$

8. 5 Registri

Binarno informacijo shranjujemo v urejenih skupinah spominskih celic, ki jim pravimo registri.

Skupina "n" spominskih celic (pravimo ji n-bitni register) je torej zmožna shraniti "n" bitov.

Operacije, ki jih izvaja nek register (vpis, brisanje, pomik), določajo zunanji kontrolni signali.

Po obsežnosti vezja, ki tvori tak register, so to vezja srednje stopnje integracije (MSI).

8. 5. 1 Pomnilniški registri

Za realizacijo lahko uporabimo kombinirano spominsko celico D - RS :

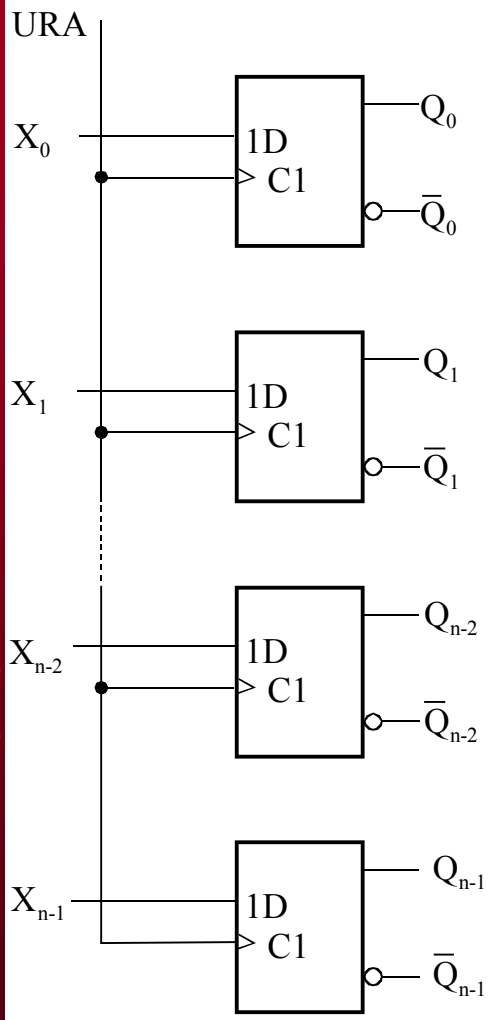
$Q_i = \overline{R}\Delta^{-1}x_i$ oziroma $\Delta^1 Q_i = \overline{R}x_i;$ $i = 1,2,...,n$

Uporaba celic RS da naslednjo enačbo stanja registra.

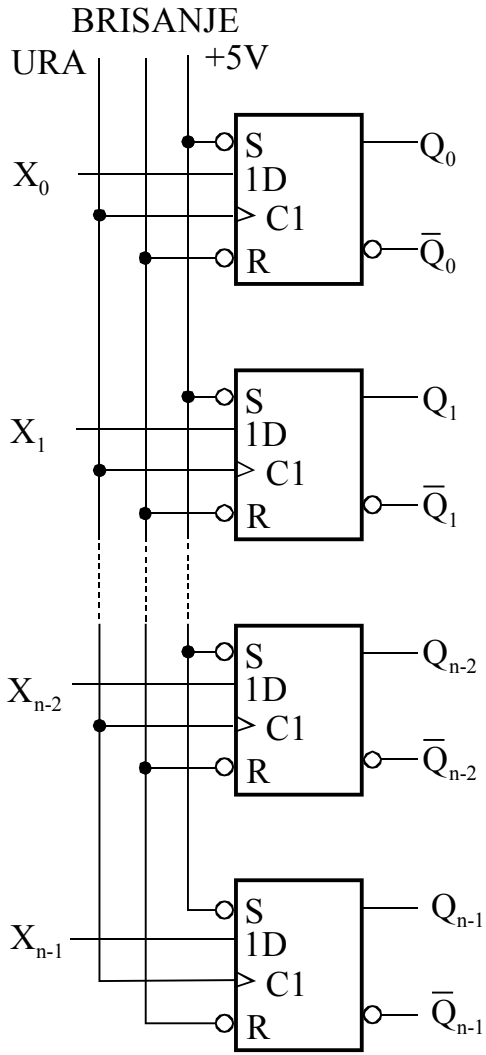
$Q_i = \Delta^{-1}(\overline{R}Q_i + x_i)$; oziroma $\Delta^1 Q_i = (\overline{R}Q_i + x_i)$ $i = 1,2,...,n$

Enačba stanj registra je torej odvisna od vrste uporabljenih spominskih celic in jo lahko priredimo potrebam pomnjenja.

Zgled:



a)



b)

$Q_i = (\Delta^{-1} x_i)$, oziroma $\Delta^1 Q_i = x_i$

$Q_i = \overline{R}(\Delta^{-1} x_i)$, oziroma $\Delta^1 Q_i = \overline{R} x_i$

$Q_i = \overline{R}(\Delta^{-1} x_i + \Delta^{-1} Q_i)$

8. 5. 2 Univerzalni premikalni register

- paralelni zapis
- serijski premik desno in
- serijski pomik levo

a) P a r a l e l n i z a p i s

Če je $P = 1$, $S_R = 0$, $S_L = 0$ se podatki vhodov x_0–x_3 prenesejo v A_0–A_3

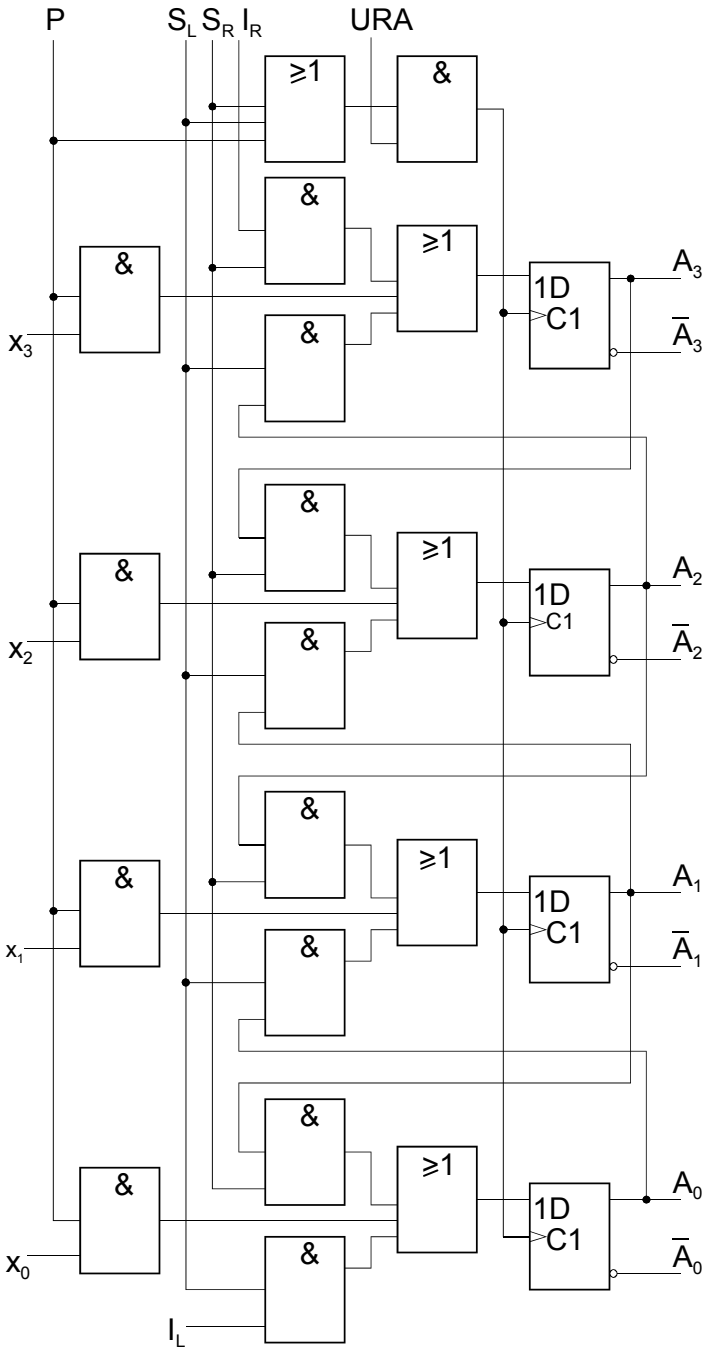
b) P o m i k d e s n o

Če postavimo $S_R = 1$, $P = 0$, $S_L = 0$, se podatki v registru pomikajo proti desni

c) P o m i k l e v o

Če je $S_L = 1$, $P = 0$, $S_R = 0$, gredo podatki proti levi.

Simbolični diagram:



$$D_{A_0} = I_L S_L + x_0 P + A_1 S_R$$

$$D_{A_1} = A_0 S_L + x_1 P + A_2 S_R$$

$$D_{A_2} = A_1 S_L + x_2 P + A_3 S_R$$

$$D_{A_3} = A_2 S_L + x_3 P + I_R S_R$$

Tabela prehajanja stanj univerzalnega registra pri izvajanju operacije pomika v levo:

S_L=1, S_R=0, P=0

Predpostavka: vhod I_L se v času izvajanja operacije ne spreminja

Sedanje stanje				Naslednje stanje I_L=1				Naslednje stanje I_L=0			
A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	1	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	1	1	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0	0
0	1	0	1	1	0	1	1	1	0	1	0
0	1	1	0	1	1	0	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1	0	0	0	0
1	0	0	1	0	0	1	1	0	0	1	0
1	0	1	0	0	1	0	1	0	1	0	0
1	0	1	1	0	1	1	1	0	1	1	0
1	1	0	0	1	0	0	1	1	0	0	0
1	1	0	1	1	0	1	1	1	0	1	0
1	1	1	0	1	1	0	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	0

Na “i”- tem mestu več bitnega registra bo imel vhod D naslednjo vhodno funkcijo:

$$D_i = Px_i + S_R Q_{i+1} + S_L Q_{i-1}; \qquad PS_R S_L = 0$$

$$D_{n-1} = Px_{n-1} + S_R I_R + S_L Q_{n-2}; \quad PS_R S_L = 0$$

$$D_0 = Px_0 + S_R Q_1 + S_L I_L; \qquad PS_R S_L = 0$$

$$\Delta^1 Q_i = Px_i + S_R Q_{i+1} + S_L Q_{i-1}$$

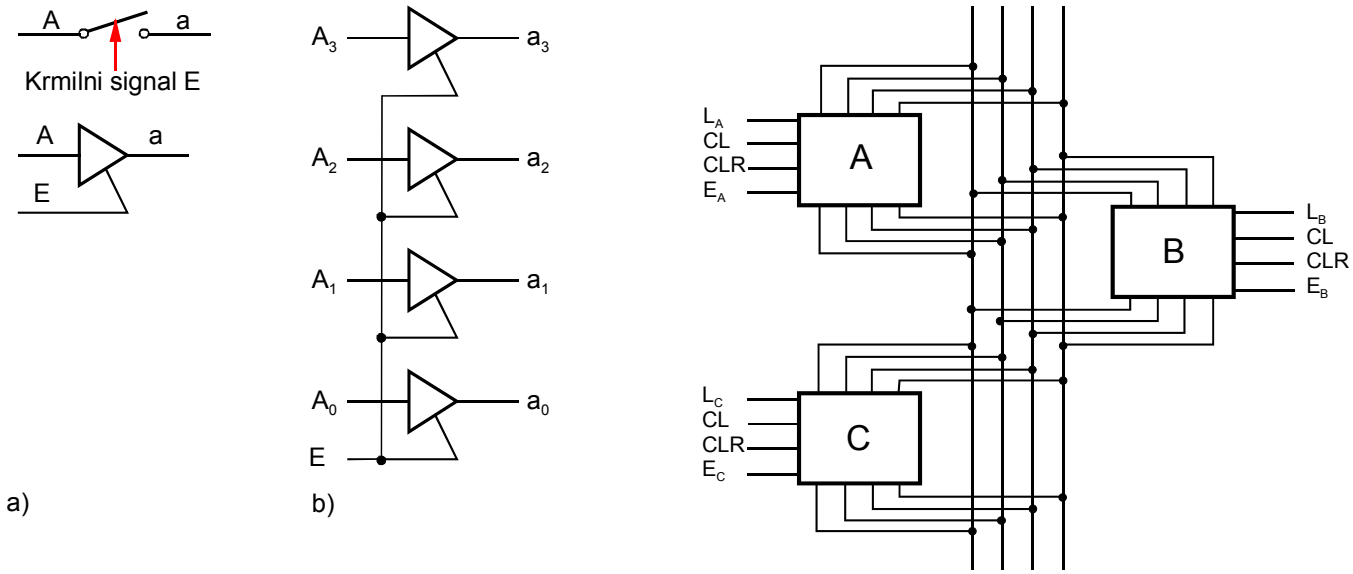
Krmiljenje premika v tem registru zahteva:

Paralelni zapis: P=1, S_R=0, S_L=0

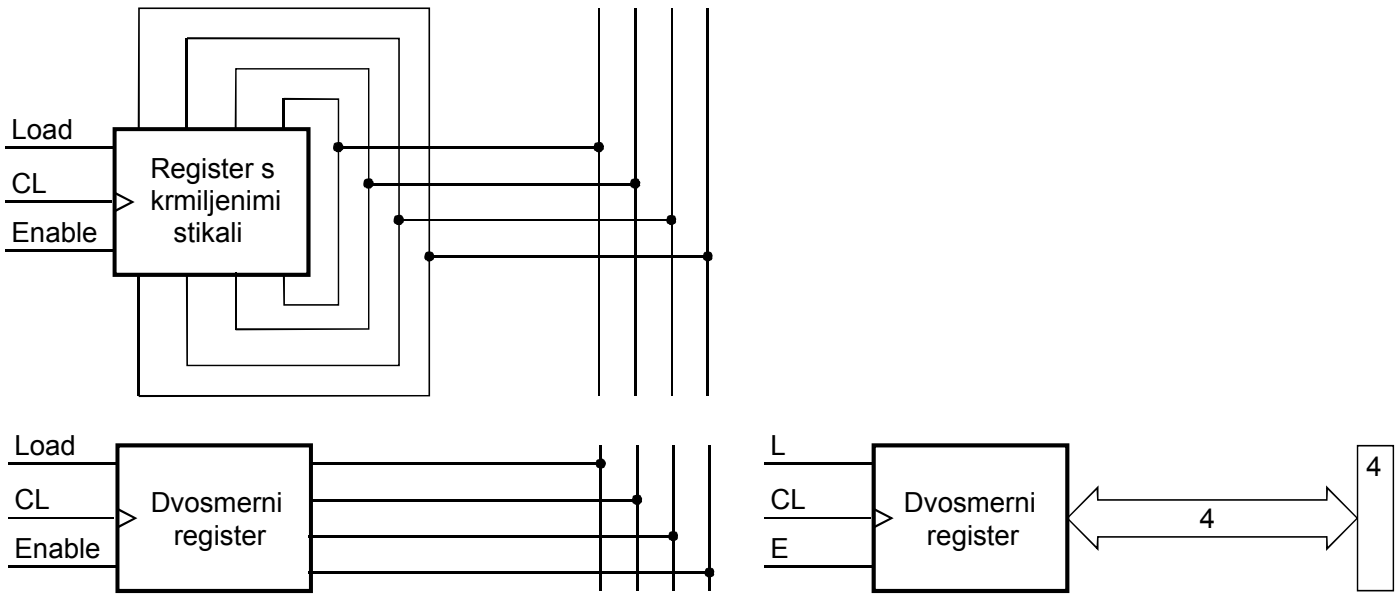
Pomik v desno: S_R=1, S_L==0, P=0

Pomik v levo: S_L=1, S_R=0, P=0

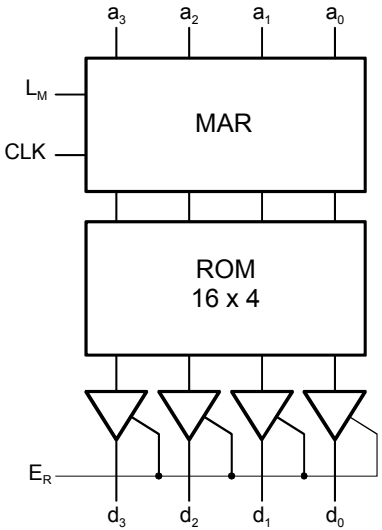
8. 5. 3 Stikalo treh stanj in povezovanje registrov



Dvosmerni registri:

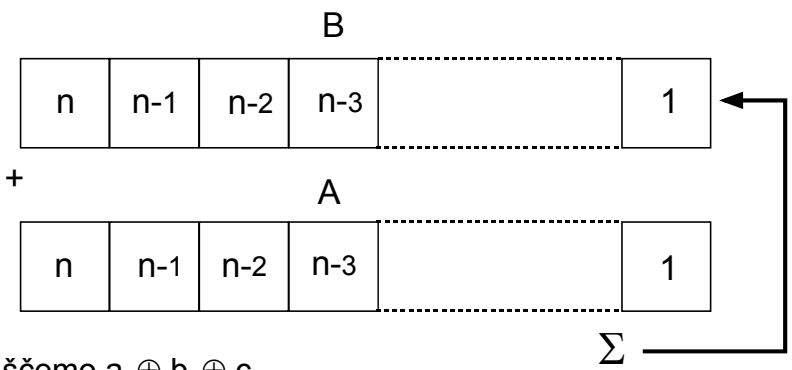


8. 5. 4 Naslavljanje ROM pomnilnika preko adresnega registra



8. 6 Sekvenčna aritmetična vezja

8. 6. 1 Seštevalniki



Iščemo $a_i \oplus b_i \oplus c_i$

a_i	b_i	c_i	$\Delta^1 q_i$	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\Delta^1 q_i = \Delta^1 b_i$$

$$\Delta^1 b_i = \overline{a_i}\,\overline{b_i} c_i + \overline{a_i} b_i \overline{c_i} + a_i \overline{b_i}\,\overline{c_i} + a_i b_i c_i$$

$$c_i = a_{i-1} b_{i-1} + a_{i-1} c_{i-1} + b_{i-1} c_{i-1}$$

$$c_1 = 0$$

Primerjava $\Delta^1 b_i$ s splošno pomnilno enačbo nam da:

$$g_{i1} = \overline{a_i}\,\overline{c_i} + a_i c_i = a_i \equiv c_i$$

$$g_{i2} = \overline{a_i} c_i + a_i \overline{c_i} = a_i \oplus c_i$$

$$\Delta^1 q_i = g_{i1} q_i + g_{i2} \overline{q_i}$$

$$D_i = (\overline{a_i \oplus c_i}) b_i + (a_i \oplus c_i) \overline{b_i}$$

$$c_1 = 0$$

$$D_1 = \overline{a_1} b_1 + a_1 \overline{b_1}$$

$$c_2 = a_1 b_1$$

$$D_2 = (\overline{a_2 \oplus a_1 b_1}) b_2 + (a_2 \oplus a_1 b_1) \overline{b_2}$$

$$c_3 = a_2 b_2 + (a_2 + b_2) a_1 b_1$$

$$D_3 = \overline{\{a_3 \oplus [a_2 b_2 + (a_2 + b_2) a_1 b_1]\}} b_3 + \{a_3 \oplus [a_2 b_2 + (a_2 + b_2) a_1 b_1]\} \overline{b_3}$$

8. 6. 2 Primerjalniki

	A= a_n, a_{n-1},.....a_0 B= b_n, b_{n-1},.....b_0
Vsebina A večja	√ √
Vsebina B večja	√
Vsebini enaki	√

Primerjavo izvajamo na mestu "n", rezultat pa dobimo na mestu n+1.

Za izvajanje serijske primerjave potrebujemo tri spremenljivke a_n, b_n in P.

P = 0 – izvajanje primerjave

P = 1 – podajanje rezultata na mestu n+1.

q_a	q_b	Pomen	f_1	f_2
0	0	A = B	0	0
0	1	A < B	0	1
1	0	A > B	1	0
1	1	X	1	1

Funkciji odvzamemo, ko je P=1

Začetno stanje q_a, q_b naj bo: $q_a = q_b = 0$

q_a	q_b	P	a_n	b_n	$\Delta^1 q_a$	$\Delta^1 q_b$	f_1	f_2
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	0	0	0
0	1	0	0	0	0	1	1	1
0	1	0	0	1	0	1	1	1

q_a	q_b	P	a_n	b_n	$\Delta^1 q_a$	$\Delta^1 q_b$	f_1	f_2
0	1	0	1	0	1	0	1	1
0	1	0	1	1	0	1	1	1
0	1	1	0	0	0	0	0	1
0	1	1	0	1	0	0	0	1
0	1	1	1	0	0	0	1	0
0	1	1	1	1	0	0	0	1
1	0	0	0	0	1	0	1	1
1	0	0	0	1	0	1	1	1
1	0	0	1	0	1	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	0	0	0	0	1	0
1	0	1	0	1	0	0	0	1
1	0	1	1	0	0	0	1	0
1	0	1	1	1	0	0	1	0
1	1	0	0	0	x	x	x	x
1	1	0	0	1	x	x	x	x
1	1	0	1	0	x	x	x	x
1	1	0	1	1	x	x	x	x
1	1	1	0	0	x	x	x	x
1	1	1	0	1	x	x	x	x
1	1	1	1	0	x	x	x	x
1	1	1	1	1	x	x	x	x

q_b \ q_a	XX	XX	00	01	XX	XX	00	01
	XX	XX	00	10	XX	XX	00	01
	10	00	00	10	10	00	00	00
	10	00	00	00	01	00	00	01

b_n a_n P P

$$\Delta^1 q_a = q_a \bar{P}(a_n + \bar{b}_n) + \bar{q}_a \bar{P} a_n \bar{b}_n$$

q_b \ q_a	XX	XX	00	01	XX	XX	00	01
	XX	XX	00	10	XX	XX	00	01
	10	00	00	10	10	00	00	00
	10	00	00	00	01	00	00	01

b_n a_n P P

$$\Delta^1 q_b = q_b \bar{P}(\bar{a}_n + b_n) + \bar{q}_b \bar{P} \bar{a}_n b_n$$

Ko je P = 0 sta funkciji f_1 in f_2 vedno 1.

f_1

f_2

q_b \ q_a	XX	XX	01	11	XX	XX	01	11
	XX	XX	10	11	XX	XX	01	11
	11	10	10	11	11	10	00	11
	11	10	00	11	11	01	01	11

b_n a_n P P

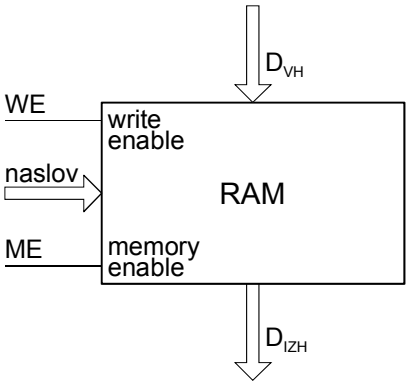
$$f_1 = \bar{P} + a_n \bar{b}_n + q_a \bar{b}_n + q_a a_n$$

q_b \ q_a	XX	XX	01	11	XX	XX	01	11
	XX	XX	10	11	XX	XX	01	11
	11	10	10	11	11	10	00	11
	11	10	00	11	11	01	01	11

b_n a_n P P

$$f_2 = \bar{P} + \bar{a}_n b_n + q_b b_n + q_b \bar{a}_n$$

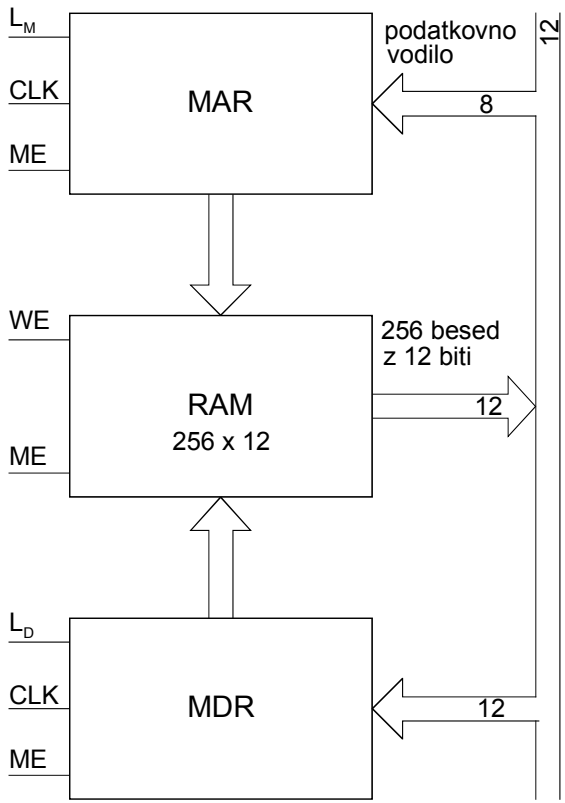
8. 7. Pomnilniška sekvenčna vezja



ME	WE	Operacija	Izhod	
0	0	zadrži	lebdeč	
0	1	zadrži	lebdeč	
1	0	beri	zvezan	naslovljena beseda na izhodu
1	1	piši	lebdeč	naslovljena beseda zapisana

Spremljajoči registri:

- spominsko adresni register MAR
- podatkovni register MDR

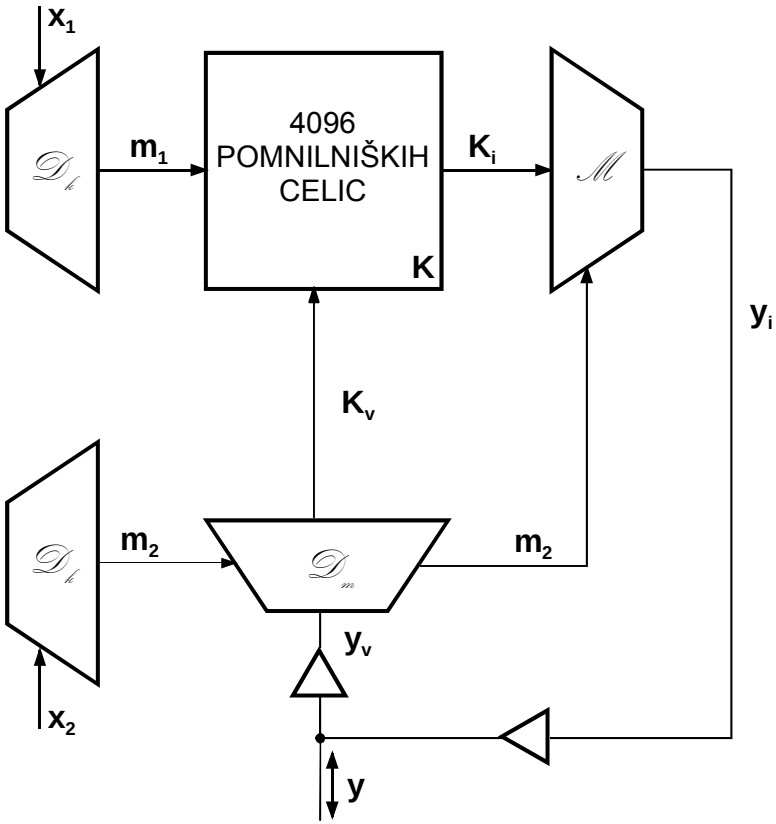


Register MDR je potreben zato, da prepreči težave pri prehodnih pojavih ob vpisovanju podatkovnih besed.

Tako je npr. MDR napolnjen, še preden postane WE=1, prav tako pa ostane podatkovna beseda še v MDR, ko pade WE ponovno na nič.

Polnjenje MDR je možno le, če je L_D = 1.

Za podrobnejše razumevanje funkcije tega pomnilnika si oglejmo še njegove osnovne gradnike.



Vpis:

$$\mathbf{K}_v = \mathbf{m}_2^T \Sigma \,\&\, \mathbf{y}_v = \mathbf{m}_1^T \Sigma \,\&\, \mathbf{y}$$

Branje pomnilnika:

$$\mathbf{y}_i = \mathbf{m}_2 \Sigma \,\&\, \mathbf{K}_i = \mathbf{m}_2 \Sigma \,\&\, (\mathbf{m}_1^T \Sigma \,\&\, \mathbf{y})$$

Pri tem je:

y – podatkovni vektor v pomnilniku

$\mathbf{y}_i$ – izhodni podatkovni vektor

$\mathbf{y}_v$ – vhodni podatkovni vektor

$\mathbf{K}_v$ – vhodna matrika

$\mathbf{K}_i$ – izhodna matrika

$\mathbf{x} = [\mathbf{x}_1 , \mathbf{x}_2]$;

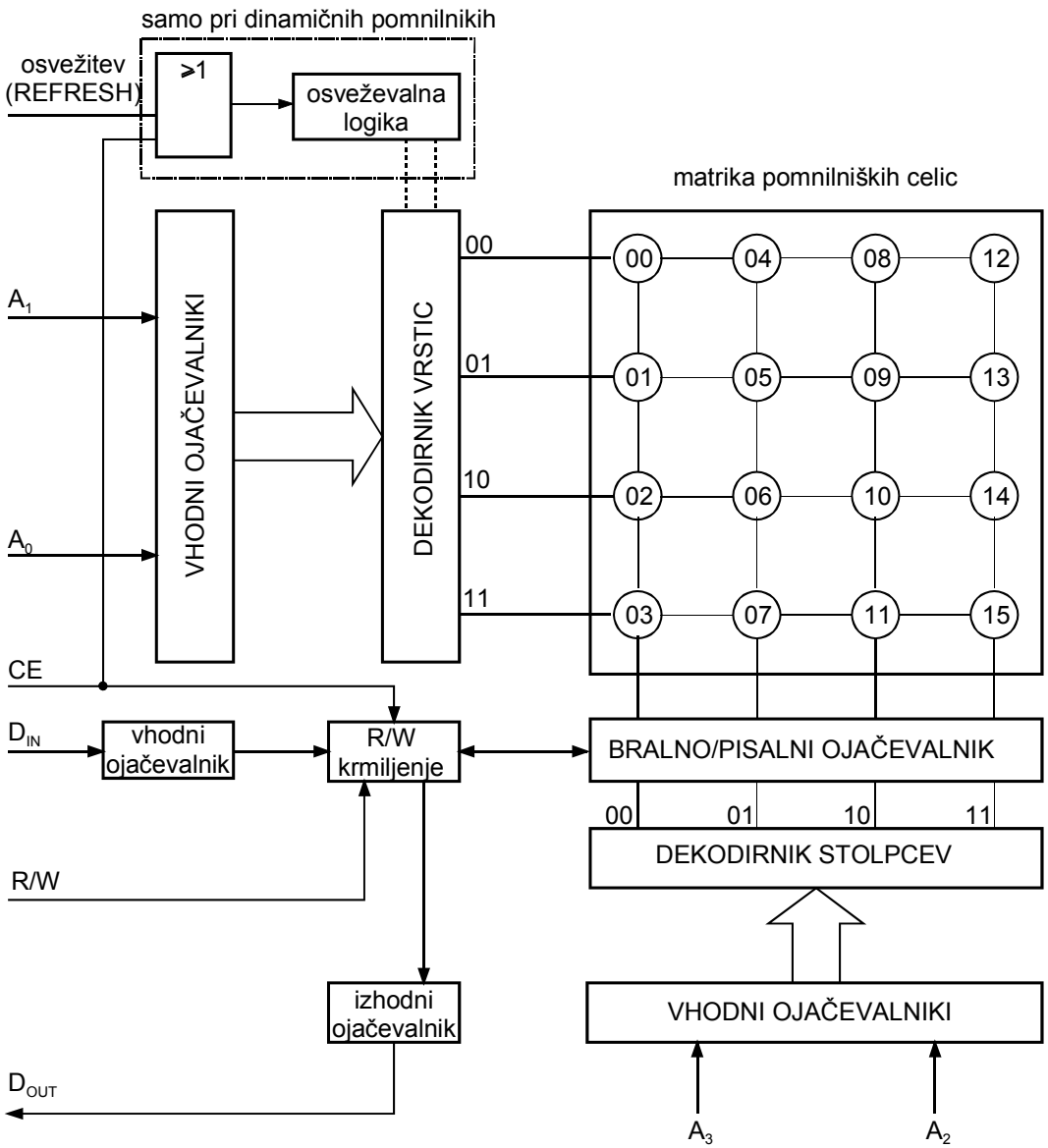
$\mathbf{m} = [\mathbf{m}_1 , \mathbf{m}_2]$

$$\mathbf{K} = \mathbf{m}^T \Sigma \,\&\, \mathbf{y}$$

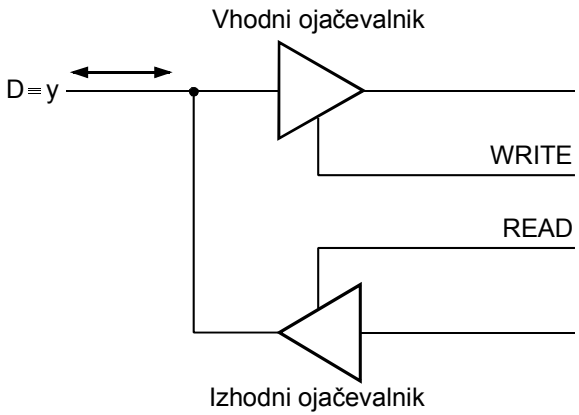
$$\mathbf{y} = \mathbf{m} \Sigma \,\&\, \mathbf{K}$$

Notranje zgradbe in načini naslavljanja RAM pomnilnikov se lahko močno razlikujejo med seboj.

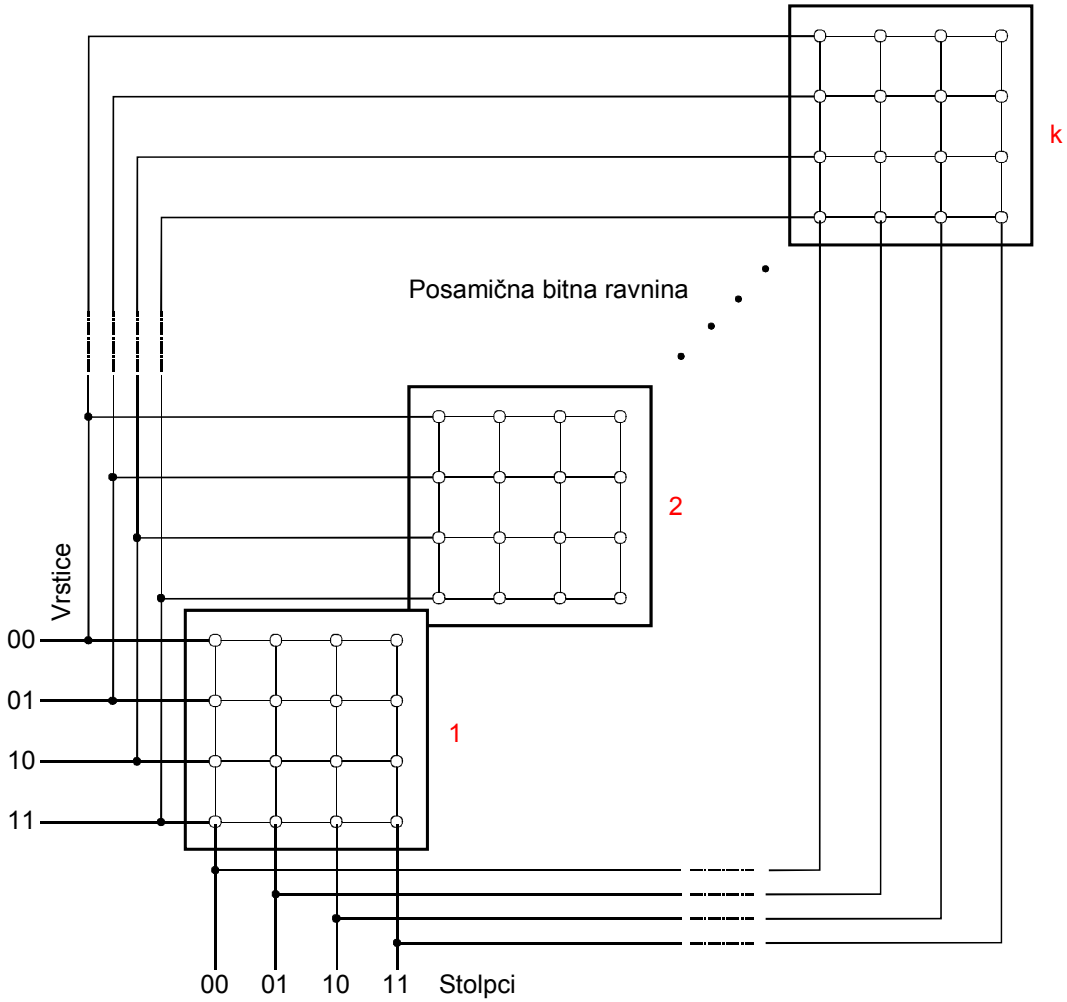
Ena od značilnih notranjih zgradb RAM pomnilnika je prikazana na spodnji sliki. Lahko je realizirana kot statični ali dinamični RAM pomnilnik. Ker dinamični pomnilnik potrebuje za posamezni bit le en tranzistor, statični pa vsaj pet, je gostota pomnilnih celic pri njem bistveno večja; potrebuje pa stalno “osveževanje”; to je obnavljanje shranjene vsebine.



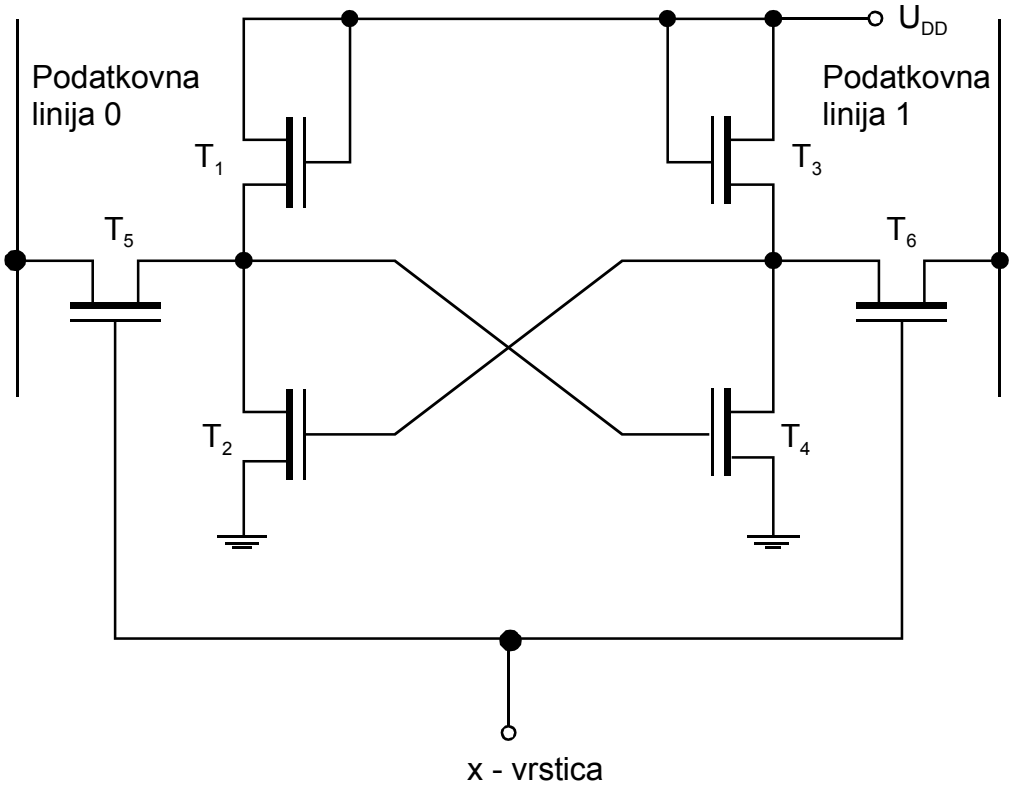
Bidirekcionalno stikalo treh stanj



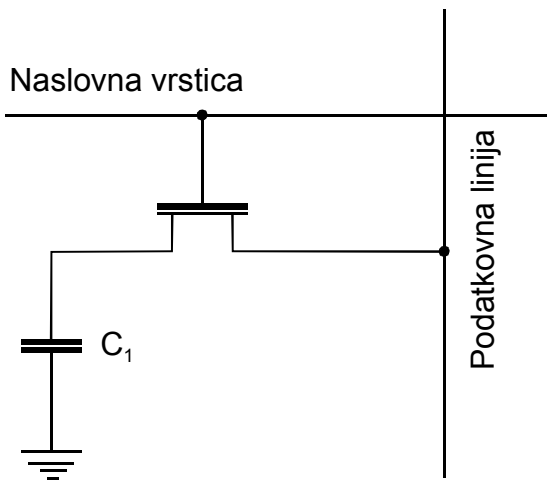
Za shranjevanje večbitnih podatkov, na primer posameznih besed, potrebujemo paralelno vključena takšna matrična polja spominskih celic, ki so bila predstavljena na prejšnji sliki. Vsaka posamezna takoimenovana bitna ravnina ima skupne naslovne linije in na njej se nahaja posamezni bit celotne besede, ki je shranjena v pomnilniku. Povezavo bitnih ravnin nazorno prikazuje naslednja slika:



Osnovna spominska celica statičnega RAM pomnilnika



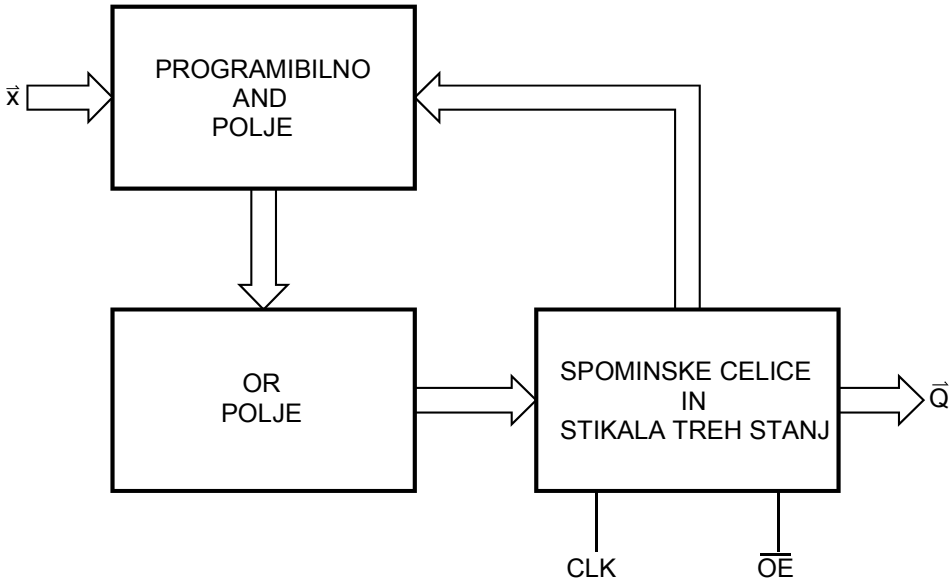
Osnovna spominska celica dinamičnega RAM pomnilnika



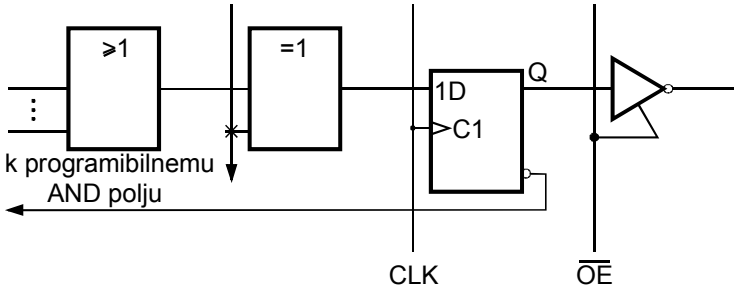
8. 8 Programibilna sekvenčna vezja

8. 8. 1 PAL arhitektura z dodatnimi spominskimi celicami

- programibilni vhodno/izhodni priključki
- programibilna polariteta izhodnih signalov
- vgrajeni XOR elementi med AND-OR poljem in spominskimi celicami.



Programiranje izhodne polaritete



Programibilni vhod je normalno priključen na maso, zato XOR vezje deluje le kot bafer.

Izhodna polariteta je zaradi invertiranja v stikalu treh stanj obrnjena.

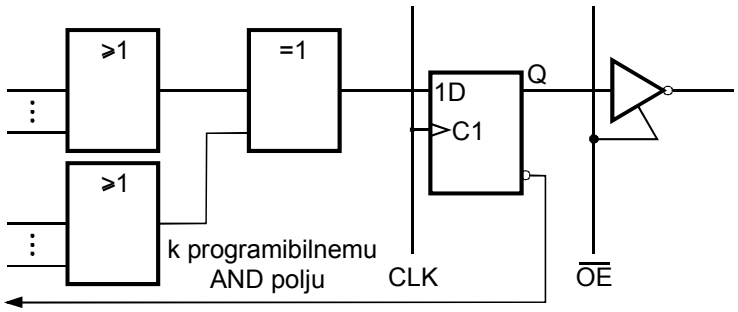
Če pa je drugi vhod v XOR sprogramiran, kar pomeni, da je zveza z maso prekinjena, je ta vhod v zraku in XOR vezje opravlja invertiranje vhodnega signala.

Izhod iz stikala treh stanj pa je v tem primeru neinvertiran.

Programibilna izhodna polariteta ni ugodna samo navzven, temveč tudi za načrtovanje znotraj vezja – enostavnejša dopolnilna funkcija.

V takem primeru je smiselno preveriti, če nima morda negirana funkcija manjše število konjunkcij, tako da to ne predstavlja več omejitve glede števila vhodov v OR vezje; izhod pa potem invertirati s XOR vezjem.

Poleg PAL vezij s programibilno izhodno polariteto obstajajo tudi vezja, ki imajo sicer na istem mestu vgrajene XOR elemente, vendar vse vhode teh elementov fiksno vezane na izhode OR vezij, tako da omogočajo funkcijsko zvezo: AND-OR-XOR.

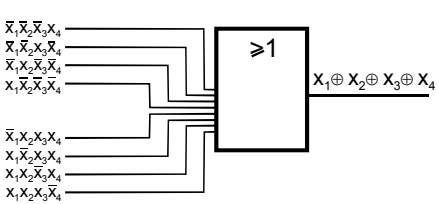


Prednost, ki jo prinaša arhitektura z vgrajenim XOR vezjem, si oglejmo na primeru realizacije XOR operacije med štirimi neodvisnimi spremenljivkami:

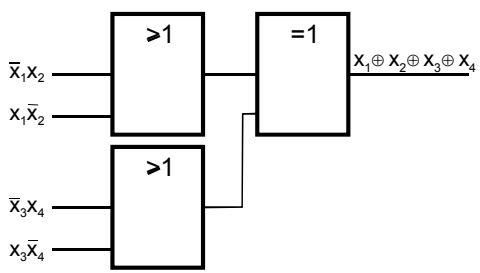
$$x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

Pri realizaciji z navadnim PAL vezjem bi potrebovali osemvhodni OR element.

Če pa uporabimo AND-OR-XOR izvedbo, pa zadostujeta dva dvovhodna OR elementa.



a)



b)

8. 8. 2 Programibilna vezja z makro celicami

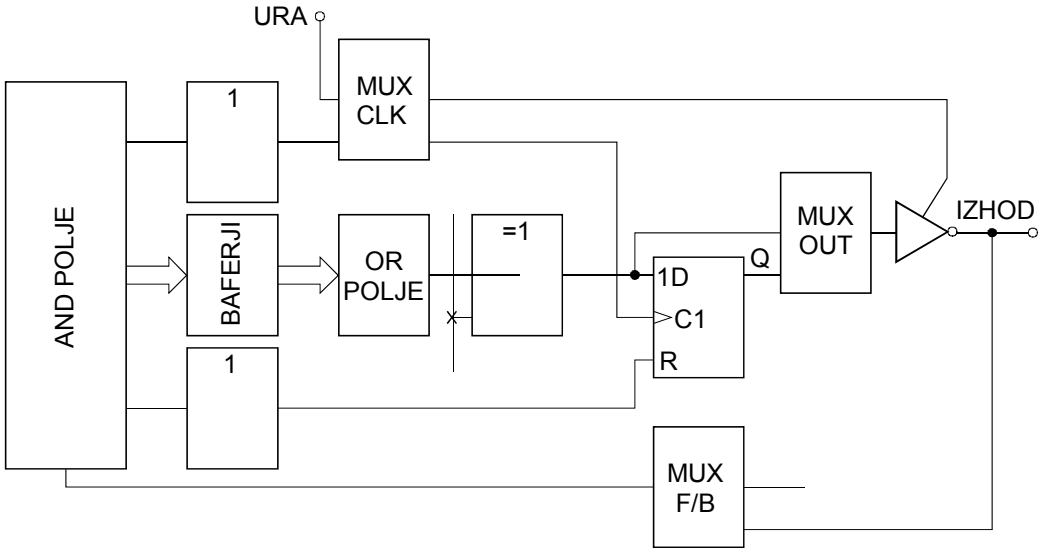
Pri teh vezjih je, za razliko od prejšnjih, ki so najbolj pogosto narejena v TTL tehnologiji, uporabljena CMOS tehnologija EEPROM vezij.

V tej tehnologiji je zelo enostavno realizirati vezja multipleksorjev, ti pa so poleg spominskih celic glavni funkcionalni sklopi makro celic.

Uporaba EEPROM tehnologije in vključitev multipleksorjev pa omogoča tem vezjem dve zelo važni prednosti:

Najprej je zaradi tehnološke izvedbe možno enostavno preprogramiranje in večja kompleksnost, ker odpade okenček za brisanje.

Makro celica firme ALTERA



Multipleksorji omogočajo neodvisno programiranje izhodnih linij, povratnih zvez in urinih impulzov za vsako celico posebej.

Selekcijski vhodi multipleksorjev so krmiljeni z izhodi EPROM vezja.

Izhodni multipleksor določa, ali bo izhod sekvenčen ali kombinacijski

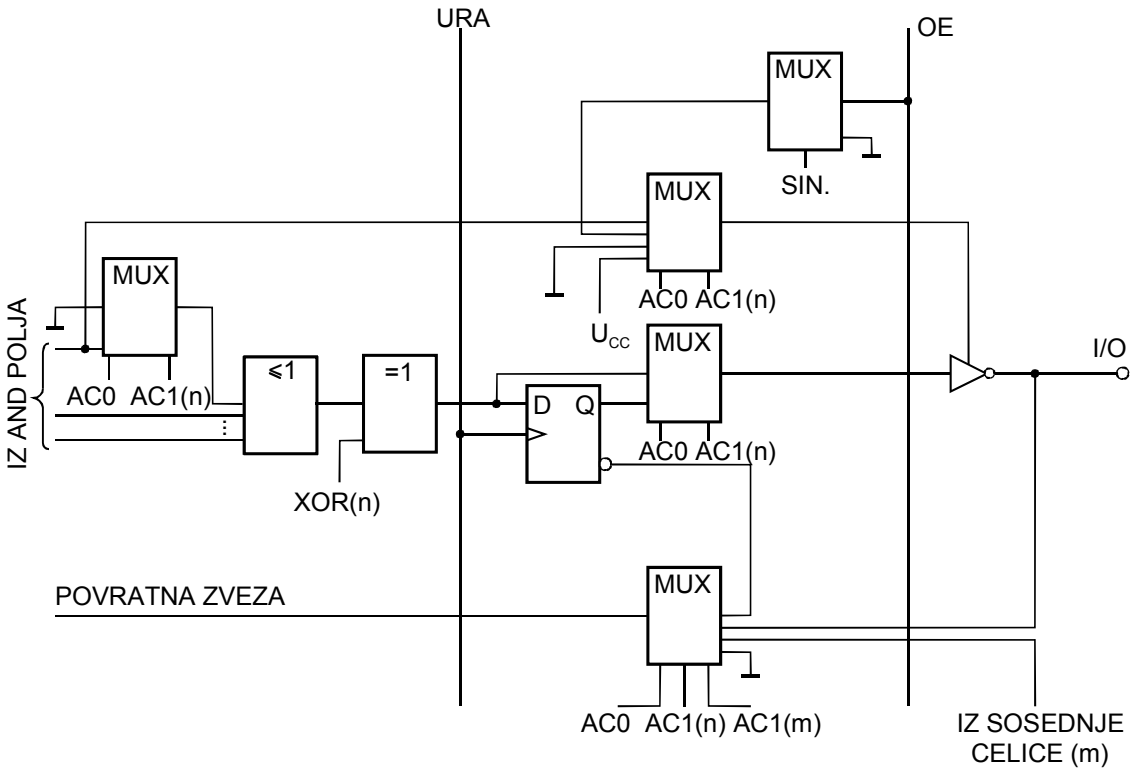
Multipleksor v povratni zvezi odloča o tem, ali bo vhod v AND polje priključen na izhod spominske celice ali pa na zunanji signal.

Na ta način lahko programiramo zunanje priključke tako, da so bodisi vhodi ali pa izhodi zopet za vsako celico posebej

Nekatere izvedbe imajo celo v vsaki celici po dve povratni zvezi.

S tem lahko izkoristimo spominsko celico za notranje stanje, pripadajoč priključek pa kot neodvisen vhod, kar bistveno prispeva k izkoriščenosti vezja.

Tretji multipeksor omogoča izbiro urinega impulza med od zunaj dovedenim ali generiranim znotraj makro celice. Ker je ta drugi urin impulz generiran z neko konjunkcijo, je lahko poljubna kombinacija neodvisnih vhodov in dodatnih zunanjih urinih signalov.



Opisani makro celici predstavljata zelo elementarno izvedbo in sta uporabni za realizacijo enostavnih avtomatov oziroma preklopnih struktur. Sodobnejše makro celice so seveda bistveno kompleksnejše in kot takšne omogočajo projektiranje zelo obsežnih in v funkcionalnem smislu zahtevnih struktur.

Makrocelica firme Xilinx iz družine XC4000