1. OSNOVNI POJMI INFORMATIKE IN KODIRANJA

Abeceda

je končna neprazna množica alfa-numeričnih simbolov, kjer je simbol nedeljiva enota: $Q = n^k$

Q - število besed, različnih dolžin, ki jih sestavlja določeno število simbolov
n - število simbolov
k - dolžina besed

Jezik

je množica smiselnih besed, ki jih tvorimo s pomočjo abecede: $J \leq A'$; v jeziku srečamo dva pojma:
- **sintaksa** – disciplina, ki proučuje strukturo jezika (jezikovna pravila)
- **semantika** – vsebinska pravila

Kodiranje

Če vsakemu elementu iz množice B pridružimo besedo, ki jo sestavljajo elementi iz množice A, govorimo o kodiranju množice B z elementi množice A. Dobljena podmnožica je kod informacije B v abecedi A.

množica $B = (b_1, b_2, \ldots\ldots b_n)$

kodiranje ↑ ↓ dekodiranje

množica $A = (a_1, a_2, \ldots\ldots a_n)$

Primer kodiranja letnice z rimskimi števili:

1 9 9 7

↓ ↓ ↓ ↓

M CM XC VII = MCMXCVII

Rimska števila - nepozicijski številski sistem:
X = 10
L = 50
C = 100
D = 500
M = 1000

Pravila kodiranja imenujemo kodiranje, novi zapis informacije pa kod.

- enakomerni kod: vse besede, ki sestavljajo kod, imajo enako število elementov oz. mest: $n \leq m^q$
  n – število znakov v kodirani besedi
  m – število elementov kodirne množice
  q – dolžina besede

- popolni enakomerni kod dobimo, če enakomerni kod vsebuje vse besede dolžine q v abecedi A
  $n = m^q$

- predstavitev informacije: podatek → zapis informacije → kodiranje informacije

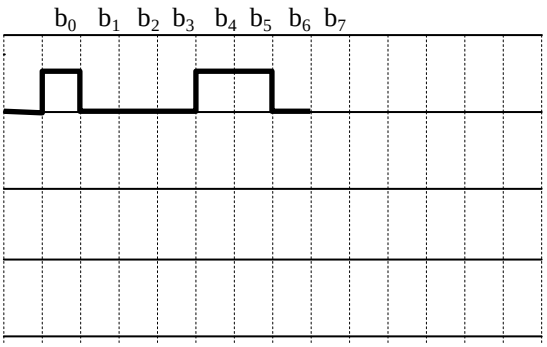
- **BCD kod** (binary coded decimal: $b_3\, b_2\, b_1\, b_0$) je ciklični enokoračni kod, ki ga sestavlja štiri oz. pet bitov. Imamo več tipov BCD kod (8421, Gray, Petheric, Lorenzo,…)

- **ASCII kod** (American Standard Code for Information Interchange) je sedembitni kod za kodiranje alfanumeričnih znakov in krmilnih ukazov. Standard se uporablja v računalniško – procesorskih sistemih za obdelavo in prenos informacij oz. podatkov.

  Značilnosti ASCI koda: $b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$
  vrednost koda: A < B < C …
  (A, B, …Z) < (a,b, ….z)
  (0, 1. 2, ...9) < (A, B, ….a, b,….)
  (krmilne funkcije: NUL, …) < SP, 0, 1, …..Z)

Primer:
Število 1997 zapisano v ASCII kodu in prikazano s časovnim diagramom!

1	=	011 0001
9	=	011 1001
9	=	011 1001
7	=	011 0111



1.1 Osnovni številski sistemi

Številske sisteme delimo:
- ne-pozicijski številski sistemi (rimska števila)
- pozicijski številski sistemi (desetiški, dvojiški, osmiški, šestnajstiški, …)

Pozicijski številski sistemi

Značilnost pozicijskih številskih sistemov je, da je vrednost števila odvisna od mesta cifre v številskem zapisu.

Primer zapisa: $1997_{(10)} = 1.10^3 + 9.10^2 + 9.10^1 + 7.10^0 = 1000 + 900 + 90 + 7$

Splošni zapis vsakega pozitivnega in realnega števila:

$$N_{(B)} = \Sigma C_i . B^i = C_n . B^n + C_{n-1} . B^{n-1} + \ldots\ldots + C_0 . B^0 + C_{-1} . B^{-1} + C_{-2} . B^{-2} + \ldots . + C_{-m} . B^{-m} =$$
$$= C_n C_{n-1} \ldots C_0, C_{-1} C_{-2} \ldots . C_{-m}$$

C – cifra številskega sistema, pri čemer velja: $0 \leq C_i < B$
B – baza ali osnova številskega sistema

Najpogosteje uporabljani pozicijski sistemi v digitalni tehniki:
- dvojiški ali binarni sistem: $11100_{(2)} = 28_{(10)};\ C \in (0,1);\ B = 2$
- osmiški ali oktalni sistem: $34_{(8)} = 28_{(10)};\ C \in (0,1,2,\ldots 7);\ B = 8$
- šestnajstiški ali heksadecimalni sistem: $1C_{(16)} = 28_{(10)};\ C \in (0,1,2,\ldots 9,A,B,C,D,E,F);\ B = 16$

Pretvorba poljubnega sistema v desetiški sistem

Poljubno število pretvorimo v desetiško število s pomočjo enačbe splošnega zapisa števila.

Primer za dvojiški sistem: $101,011_{(2)} = 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3} = 5,375_{(10)}$

Primer za osmiški sistem: $77,51_{(8)} = 7.8^1 + 7.8^0 + 5.8^{-1} + 1.8^{-2} = 63,640625_{(10)}$

Primer za šestnajstiški sistem: $CD,E8_{(16)} = 12.16^1 + 13.16^0 + 14.16^{-1} + 8.16^{-2} = 205,90625_{(10)}$

Pretvorba decimalnega števila v število drugega številskega sistema

Postopek pretvorbe celoštevilčnega dela

Celi del števila delimo z osnovo B, ostanek označimo s C_0. Postopek ponavljamo tako dolgo, dokler ne pridemo do zadnjega ostanka C_n, to je takrat, ko postane celoštevilčni del rezultata deljenja enak nič. Število zapišemo: $C_n \ldots C_{0\,(B)}$.

Primer: Število $115_{(10)}$ pretvorimo v ostale številske sisteme (2), (8), (16)!

	ostanek		ostanek		ostanek
$115_{(10)} : 2$	**1** → C_0	$115 : 8$	**3**	$115 : 16$	**3**
$= 57 : 2$	**1** → C_1	$= 14 : 8$	**6**	$= 7 : 16$	**7**
$= 28 : 2$	**0** → C_2	$= 1 : 8$	**1**	$= 0$	↑
$= 14 : 2$	**0** → C_3	$= 0$	↑		
$= 7 : 2$	**1** → C_4				
$= 3 : 2$	**1** → C_5				
$= 1 : 2$	**1** → C_6				
$= 0$	↑				

$115_{(10)} = 1110011_{(2)}$ **$115_{(10)} = 163_{(8)}$** **$115_{(10)} = 73_{(16)}$**

Postopek pretvorbe decimalnega dela

Decimalni del množimo z osnovo B. Pri množenju dobljeni celi del označimop s C_{-1}, nato preostali decimalni del zopet množimo z osnovo B in dobljeni celi del označimo s C_{-2}. Postopek ponavljamo, dokler z množenjem decimalna mesta niso enaka 0, ali dokler se ne zadovoljimo s številom mest. Zapis števila je: $0,C_{-1}C_{-2}\ldots\ldots_{(B)}$

Primer: Število $0,2_{(10)}$ pretvorimo v ostale številske sisteme (2), (8), (16)!

0,2 * 2 = **0**,4 ↓	0,2 * 8 = **1**,6 ↓	0,2 * 16 = **3**,2 ↓
0,4 * 2 = **0**,8	0,6 * 8 = **4**,8	0,2 * 16 = **3**,2
0,8 * 2 = **1**,6	0,8 * 8 = **6**,4	0,2 * 16 = **3**,2
0,6 * 2 = **1**,2	0,4 * 8 = **3**,2	0,2 * 16 = **3**,2
0,2 * 2 = **0**,4		
0,4 * 2 = **0**,8		
0,8 * 2 = **1**,6		
0,6 * 2 = **1**,2		

$0,2_{(10)} = 0,$**00110011**$_{(2)}$ $0,2_{(10)} = 0,$**1463**$_{(8)}$ $0,2_{(10)} = 0,$**333**$\ldots._{(16)}$

Pretvorba med (2), (8) in (16) številskimi sistemi

Števila lahko medsebojno pretvarjamo med (2), (8) in (16) številskimi sistemi tako, da pri pretvorbi dvojiškega in osmiškega sistema upoštevamo tri-bitni zapis števila, pri pretvorbi dvojiškega in šestnajstiškega sistema pa upoštevamo štiri-bitni zapis števila.

Primer:

$163_{(8)} = 001\ 110\ 011_{(2)} = 0111\ 0011_{(2)} = 73_{(16)}$

1.2. Predznačena števila – števila s predznakom

N ≥ 0 … pozitivno število (+)
N < 0 … negativno število (-)

Tipi podatkov:	byte	8 bitov
	word	16 bitov
	long word	32 bitov
	quadword	64 bitov
	octaword	128 bitov

pozitivno število: **0**0101101
negativno število: **1**0100011

Klasični zapis števila je zapis s fiksno vejico, pri katerem velja, da informacijo o predznaku nosi bit MSB (most significant bit):

MSB = 0 … število je pozitivno
MSB = 1 … število je negativno

Enojni komplement števila

je dopolnitev danega števila do števila največje vrednosti določenega številskega sestava. Izračunamo ga tako, da določimo razliko med cifro najvišje vrednosti in dano cifro. Enojni komplement števila določimo z enojnimi komplementi posameznih cifer v številu. Za binarna števila določimo enojni komplement enostavneje tako, da zamenjamo ničle z enicami in obratno.

Primer: Za dana števila zapišimo enojne komplemente!

	dvojiški zapis	osmiški zapis	desetiški zapis	šestnajstiški zapis
max.vrednost:	1111 1111	7777	9999	FFFF
dano število:	0111 0100	1245	1997	3E54
enojni kompl.:	1000 1011	6532	8002	C1AB

Dvojni komplement števila

je za ena povečan njegov enojni komplement. Dobimo ga s prištevanjem enice na zadnjem mestu števila LSB (last significant bit). Lastnost dvojnega komplementa danega števila je v tem, da skupaj z danim številom da vsoto nič, kar pomeni dve enako veliki toda nasprotno pred-značeni števili.

Primer 1: Od števila 3E54 $_{(16)}$ odštejmo število 1234 $_{(16)}$ po metodi dvojnega komplementa in izračun preverimo tudi v destiškem sistemu!

	F F F F			
dano število:	1 2 3 4		izračun:	3 E 5 4
enojni komplement:	E DCB			+ EDCC
	+ 1			1 **2C2 0**
dvojni komplement:	EDCC (negativno število)			

Pakirana decimalna števila
Vsako desetiško število je kodirano v BCD 8421 kodu, za predznak so uporabljeni najnižji 4 biti v zadnjem byte-u.
Zapis predznaka: C ⇔ '+' = 1100 in D ⇔ '-' = 1101

Zapis števila: xxxx xxxx xxxx xxxx xxxx yyyy
d d d d d s; d – digit; s - sign

Primer: Desetiško število – 2305 $_{(10)}$ zapišimo kot pakirano decimalno število!
- 2305 $_{(10)}$ = 02 30 5D $_{(BCD\ 8421)}$ = 0000 0010 0011 0000 0101 1101 $_{(BCD\ 8421)}$

ASCII številski niz
Vsaka cifra je zapisana z enim bytom ASCII kode, prav tako predznak števila.
Zapis predznaka: 2B ⇔ '+' in 2D ⇔ '-'

Zapis števila: yyyyyyyy xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
sign digit digit digit digit d – digit; s - sign

Primer: Desetiško število – 2305 $_{(10)}$ zapišimo kot ASCII številski niz!
- 2305 $_{(10)}$ = 2D 32 33 30 35 $_{(ASCII\ številski\ niz)}$

Zapis s plavajočo vejico (normirani eksponentni zapis števila)
Število je predstavljeno z dvojiško mantiso in dvojiškim eksponentom. Normiranje je takšno, da binarna mantisa ne doseže vrednosti 1 in je vedno večja ali kvečjemu enaka $0{,}1_{(2)} = 0{,}5_{(10)}$.

Osnovna oblika zapisa s plavajočo vejico: $N = \pm\ m \cdot B^{\pm e}$
Za normirano obliko mantise velja pogoj: $B^{-1} \le m < B^0$
Za binarni številski sistem velja: $B = 2;\quad 2^{-1} \le m < 1 \Rightarrow 0{,}5 \le m < 1$
Za desetiški številski sistem velja: $B = 10;\ 10^{-1} \le m < 1 \Rightarrow 0{,}1 \le m < 1$

Primeri zapisa števil s plavajočo vejico:

število:	normiran zapis:
$9_{(10)} = 1001_{(2)}$	$0{,}9 \cdot 10^1 = 0{,}1001 \cdot 2^{+4}$
$2{,}5_{(10)} = 10{,}1_{(2)}$	$0{,}25 \cdot 10^1 = 0{,}101 \cdot 2^{+2}$
$0{,}1875_{(10)} = 0{,}0011_{(2)}$	$0{,}1875 \cdot 10^0 = 0{,}11 \cdot 2^{-2}$

Oblika normiranega zapisa z 10 byti (80 biti)

S eksponenta bit 79	Eksponent biti 78 - 64	S mantise bit 63	Mantisa biti 62 - 0
2 byta		8 bytov	

- vrednost mantise je normalno zapisana v decimalnem delu od bita 62 do bita 0
- predznak mantise je zapisan v bitu 63 (bit 63 = 1 ⇒ predznak je negativen)
- eksponent s predznakom je zapisan z dvojnim komplementom

Primer zapisa števila v obliki 10 byte-nega zapisa v normirani obliki:

$25{,}125_{(10)} = 11001{,}001_{(2)} = 0,\ 11001001 \cdot 2^{+5}$

S – E / bit 79	E biti 78 - 64	S - M / bit 63	M biti 62 - 0
0	0 ……...0101	0	110010010……..……0

2. OSNOVE DIGITALNIH KRMILNIH SISTEMOV

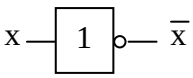
2. 1 Preklopna algebra

Preklopna algebra je matematična struktura, kjer imamo opravka z množico elementov x_1, x_2, …, ki lahko zavzamejo vrednost 0 ali 1. Nad temi elementi lahko izvajamo tri osnovne operacije: negacijo, konjunkcijo in disjunkcijo. konjunkcija je glede na disjunkcijo prednostna operacija.

Boolov podatek: 0 1 $X \in (0,1) \Rightarrow (x_1, x_2)$
ne da
false true

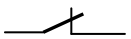
Osnovne operacije preklopne algebre

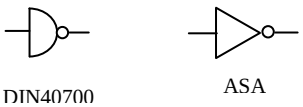
Negacija – logično nasprotovanje: $\overline{x}$, $0 \rightarrow 1$; simboli:IEC 117



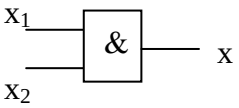
pravilnostna tabela: stikalo: mirovni kontakt releja

x	$\overline{x}$
0	1
1	0



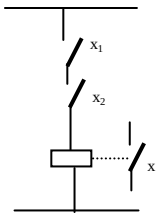


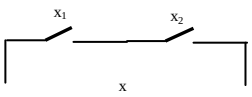
Konjunkcija – logično množenje (logični IN): Λ, ., $x = x_1 . x_2$; simbol:



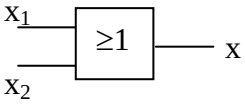
pravilnostna tabela: stikalo: zaporedna vezava kontaktov

x_1	x_2	$x_1.x_2$
0	0	0
0	1	0
1	0	0
1	1	1



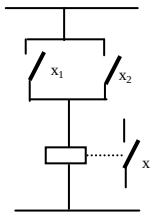


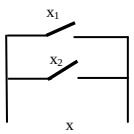
Disjunkcija – logično seštevanje (logični ALI): V, +, $x = x_1 + x_2$; simbol:



pravilnostna tabela: stikalo: vzporedna vezava kontaktov

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1





Osnovne zakonitosti preklopne algebre

- antipodnost: $x . \overline{x} = 0$ $x + \overline{x} = 1$
- komutativnost: $x_1 . x_2 = x_2 . x_1$ $x_1 + x_2 = x_2 + x_1$
- asociativnost: $x_1.x_2.x_3 = (x_1.x_2).x_3 = x_1.(x_2.x_3)$ $x_1+x_2+x_3 = (x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$
- distributivnost: $x_1(x_2 + x_3) = x_1x_2 + x_1x_3$ $x_1 + (x_2.x_3) = (x_1 + x_2) . (x_1 + x_3)$
- absorbtivnost: $x_1(x_1 + x_2) = x_1x_1 + x_1x_2 = x_1$ $x_1 + (x_1.x_2) = (x_1 + x_1) . (x_1 + x_2) = x_1$
- nevtralni element: je konstanta, ki jo priključimo izrazu, ne da bi spremenili njegov pomen
  $x_1 . 1 = x_1$; nevtralni element konjunkcije je 1
  $x_1 + 0 = x_1$; nevtralni element disjunkcije je 0

Teoremi preklopne algebre

- teoremi logičnega množenja: $x \cdot 0 = 0$; $x \cdot 1 = x$; $x \cdot x = x$; $\overline{x} \cdot x = 0$; $\overline{x} \cdot \overline{x} = \overline{x}$
- teoremi logičnega seštevanja: $x + 0 = x$; $x + 1 = 1$; $x + x = x$; $x + \overline{x} = 1$; $\overline{x} + \overline{x} = \overline{x}$
- teoremi z eno, dvema ali več spremenljivkami – priloga

- **teoremi negacije – De Morganov izrek:**

$\overline{\overline{x}} = x$; $\overline{\overline{\overline{x}}} = \overline{x}$

1. oblika izreka: $\overline{x_1 + x_2} = \overline{x}_1 \cdot \overline{x}_2$ ali

$$x_1 + x_2 = \overline{\overline{x}_1 \cdot \overline{x}_2}$$

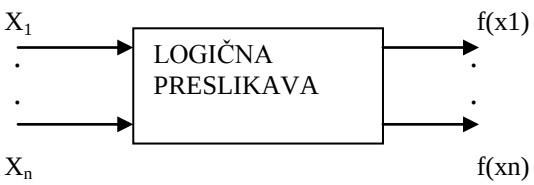
x_1	x_2	$x_1 + x_2$	$\overline{x_1 + x_2}$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 \cdot \overline{x}_2$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

2. oblika izreka: $\overline{x_1 \cdot x_2} = \overline{x}_1 + \overline{x}_2$ ali

$$x_1 \cdot x_2 = \overline{\overline{x}_1 + \overline{x}_2}$$

x_1	x_2	$x_1 \cdot x_2$	$\overline{x_1 \cdot x_2}$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 + \overline{x}_2$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

2. 2 Logične funkcije



Pravilnostna tabela:

x_1 x_2 …… x_n	f(x1….xn)
0 0 …….. 0	α_1
0 0 …….. 1	α_2
. . .	.
. . .	.
. . .	.
1 1 …….. 11	α_n

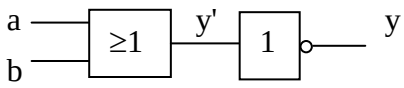
- X_n : vhodna spremenljivka
- f(x1): izhodna funkcija
- α_n : funkcijska vrednost, ki lahko zavzame vrednost 0 ali 1

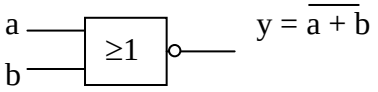
Če so X1 do Xn logične spremenljivke, tedaj je funkcija Y = f(X1…Xn) funkcija preklopne algebre ali logična funkcija. Dve logični funkciji sta enaki, če dajeta pri enakih kombinacijah spremenljivk enake funkcijske vrednosti.

Elementarne povezave logičnih funkcij

- **osnovne elementarne logične funkcije**: negacija, konjunkcija in disjunkcija
- **sestavljene elementarne logične funkcije**

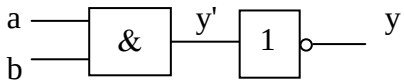
1. Pierce-ova funkcija (NEALI, NOR): $y = \overline{y'} = \overline{a + b} = \overline{a} \cdot \overline{b} = a \downarrow b$

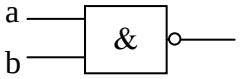




a	b	y'	y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	

2. Sheffer-jeva funkcija (NEIN, NAND): $y = \bar{y}' = \overline{a \,.\, b} = \bar{a} + \bar{b} = a \uparrow b$



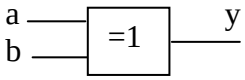


$y = \overline{a \,.\, b}$

a	b	y'	y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

- **posebne elementarne logične enačbe**

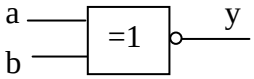
1. antivalenca (ekskluzivni ALI, EXOR)



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$y = a \oplus b = \bar{a} \,.\, b + a \,.\, \bar{b}$

2. ekvivalenca (negacija antivalence)



a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

$y = \overline{a \oplus b} = \bar{a} \,.\, \bar{b} + a \,.\, b$

2. 3 Preklopne oblike logičnih funkcij

1. Popolna disjunktivna oblika funkcije (popolna disjunktivna enačba - PDE) in mintermi

PDE je logična vsota produktov funkcijskih vrednosti α_i in mintermov m_i, kjer lahko zavzamejo funkcijske vrednosti α_i vrednost 0 ali 1. Minterm **m** je logični produkt spremenljivk, v katerem lahko spremenljivka nastopa v negirani ali nenegirani vrednosti.

Oblika PDE: $f(x_1, \ldots x_n) = \Sigma\, \alpha_i \,.\, m_i = \alpha_0 \,.\, m_0 + \alpha_1 \,.\, m_1 + \ldots\ldots + \alpha_i \,.\, m_i$

2. Popolna konjunktivna oblika funkcije (popolna konjunktivna enačba - PKE) in makstermi

PKE je logični produkt vsot, ki jih tvorijo funkcijske vrednosti β_i in makstermi M_i. Maksterm 'M' je negacija minterma in je logična vsota spremenljivk, kjer spremenljivka nastopa v negirani vrednosti glede na stanje v pravilnostni tabeli.

Oblika PKE: $f(x_1, \ldots x_n) = \Pi\, (\beta_i + M_i) = (\beta_0 + M_0) \,.\, (\beta_1 + M_1) \,.\, \ldots\ldots \,.\, (\beta_i + M_i)$

Primer: Zapis PDE in PKE na osnovi dane pravilnostne tabele

a	b	c	f (a,b,c)	mintermi	f (a,b,c)	makstermi
0	0	0	$\alpha_0 = 0$	$m_0 = \bar{a}.\bar{b}.\bar{c}$	$\beta_7 = 0$	$M_7 = \overline{\bar{a}.\bar{b}.\bar{c}} = a + b + c$
0	0	1	$\alpha_1 = 1$	$m_1 = \bar{a}.\bar{b}.c$	$\beta_6 = 1$	$M_6 = a + b + \bar{c}$
0	1	0	$\alpha_2 = 1$	$m_2 = \bar{a}.b.\bar{c}$	$\beta_5 = 1$	$M_5 = a + \bar{b} + c$
0	1	1	$\alpha_3 = 0$	$m_3 = \bar{a}.b.c$	$\beta_4 = 0$	$M_4 = a + \bar{b} + \bar{c}$
1	0	0	$\alpha_4 = 0$	$m_4 = a.\bar{b}.\bar{c}$	$\beta_3 = 0$	$M_3 = \bar{a} + b + \bar{c}$
1	0	1	$\alpha_5 = 1$	$m_5 = a.\bar{b}.c$	$\beta_2 = 1$	$M_2 = \bar{a} + b + c$
1	1	0	$\alpha_6 = 0$	$m_6 = a.b.\bar{c}$	$\beta_1 = 0$	$M_1 = \bar{a} + \bar{b} + c$
1	1	1	$\alpha_7 = 0$	$m_7 = a.b.c$	$\beta_0 = 0$	$M_0 = \bar{a} + \bar{b} + \bar{c}$

$f_{PDE} = m_1 + m_2 + m_5 = \bar{a}.\bar{b}.c + \bar{a}.b.\bar{c} + a.\bar{b}.c = V_3\,1,2,3$

$f_{PKE} = M_0 \,.\, M_1 \,.\, M_3 \,.\, M_4 \,.\, M_7 = \Lambda_3\,0,1,3,4,7$

Na osnovi pravilnostne tabele je mogoče izvesti funkcijske enačbe, ki zagotavljajo zahtevano povezavo vhodnih spremenljivk. Dobimo PDE ali PKE, zato se pravilnostna tabela razširi še za stolpca 'm' in 'M'. Mintermi so konjunktivne povezave, makstermi pa disjunktivne povezave vseh vhodnih spremenljivk.

Disjunktivna normalna oblika enačbe nastane z disjunktivno povezavo vseh mintermov s funkcijsko vrednostjo 1, konjunktivna normalna oblika pa nastane s konjunktivno povezavo vseh makstermov s funkcijsko vrednostjo 0. V primeru večjega števila funkcijskih vrednosti 0 (manj je vrednosti 1), uporabimo zapis PDE, v nasprotnem primeru pa zapis PKE.

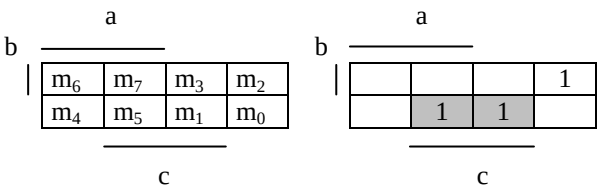
Zapis PDE – mintermov v Veitchevem diagramu

Minterm lahko zapišemo v diagram in s pomočjo združevanja posameznih mintermov v skupine po 2, 4, 8 lahko PDE minimiziramo in poenostavimo. V primeru redundantnih stanj lahko le ta uporabimo za minimizacijo.

Primer zapisa mintermov za logično enačbo:

$f_{PDE} = m_1 + m_2 + m_5 = \bar{a}.\bar{b}.c + \bar{a}.b.\bar{c} + a.\bar{b}.c$

Veitchev diagram:



- minimizirana oblika enačbe: $f_{PDE} = \bar{b}.c + \bar{a}.b.\bar{c}$

3. LOGIČNA VEZJA

Načrtovanje in projektiranje logičnih vezij zajema naslednje korake:
- definicija problema in opis tehnologije (tehnološka shema)
- določanje vhodnih spremenljivk – signalov (pritisne tipke komandnih pultov, senzorika, mikro-stikala, končna varnostna stikala,…)
- določanje izhodnih spremenljivk – kanalov (kontaktorji, motorji, pnevmatski ali hidravlični cilindri, ventili, signalizacija)
- izdelava funkcijskih logičnih enačb in pravilnostnih tabel, določanje algoritmov delovanja
- minimiziranje in optimiranje logičnih preklopnih funkcij
- izdelava funkcijskih načrtov in časovnih diagramov
- izdelava, preizkus in analiza logičnega vezja (NAND, NOR tehnologija)

Logična vezja delimo v naslednje osnovne skupine:
- osnovna kombinacijska vezja
- aritmetična vezja
- prekodirna vezja
- multiplekser in demultiplekser
- digitalna primerjalna vezja (komparator)

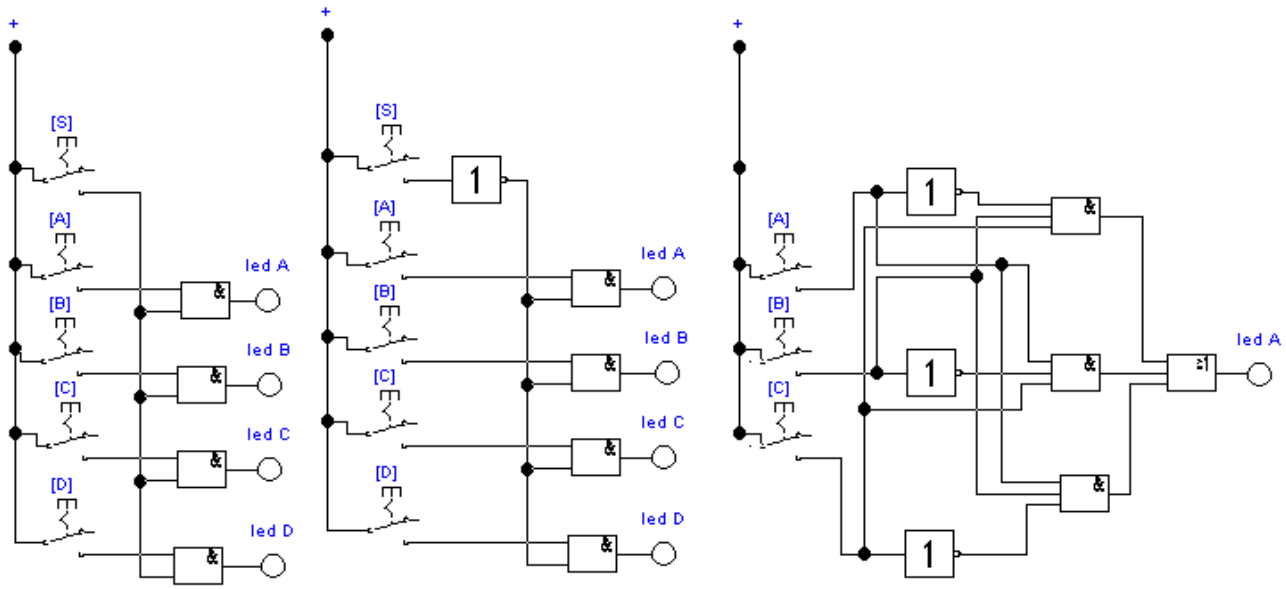
3.1 Osnovna kombinacijska vezja

a) sprostitev signala; prehod signala omogoči sprostitveni signal S. Vezje se uporablja za krmiljenje krmilnih relejev, kontrolo izhodnih signalov, …

b) zapora signala (zapahovanje); prehod signalov A, B, C, D je preprečen s signalno zaporo S. Izhodni kanali so aktivni samo pri izklopljenem stikalu S.

c) izbirno vezje x/y (2/3); ima dva ali več vhodov, na izhodu tega vezja se sme pojaviti signal 1 le tedaj, če je aktivno samo določeno število vhodnih kanalov x od skupnega števila y. Na izhodu izbirnega vezja 2/3 se sme pojaviti signal 1 le tedaj, ko sta hkrati postavljena samo 2 od treh vhodnih signalov. V vseh drugih primerih mora biti na izhodu signal 0.

a) sprostitev signala b) zapora signala c) izbirno vezje 2/3



3. 2 Aritmetična vezja

Popolni seštevalnik

Popolni seštevalnik je razširitev polovičnega seštevalnika tako, da upošteva prenos z nižjega bita C_{i-1} in ga zato lahko uporabimo za seštevanje dveh binarnih števil na katerem koli mestu binarnega števila.

Pravilnostna tabela:

a_i	b_i	C_{i-1}	S_i	C_i
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

Logične enačbe:

$$S_i = \overline{a_i} \,.\, \overline{b_i} \,.C_{i-1} + \overline{a_i} \,.\, b_i \,.\overline{C_{i-1}} + a_i \,.\, \overline{b_i} \,.\overline{C_{i-1}} + a_i \,.\, b_i \,.C_{i-1} =$$

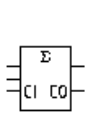
$$= \overline{C_{i-1}}.(a_i.\, \overline{b_i} + \overline{a_i} \,.b_i) + C_{i-1}.(\overline{a_i}.\, \overline{b_i} + a_i \,.b_i) = \overline{C_{i-1}}.(a_i \oplus b_i) + C_{i-1}.(\overline{a_i \oplus b_i})$$

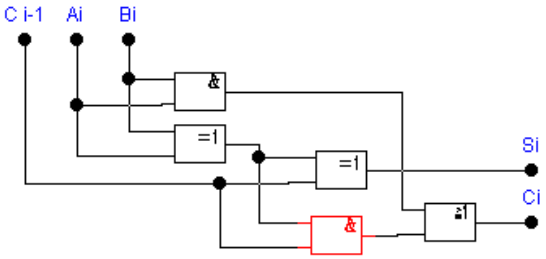
$$\mathbf{= C_{i-1} \oplus (a_i \oplus b_i)}$$

$$C_i = \overline{a_i} \,.\, b_i \,.C_{i-1} + a_i \,.\, \overline{b_i} \,.C_{i-1} + a_i \,.\, b_i \,.\overline{C_{i-1}} + a_i \,.\, b_i \,.C_{i-1} =$$

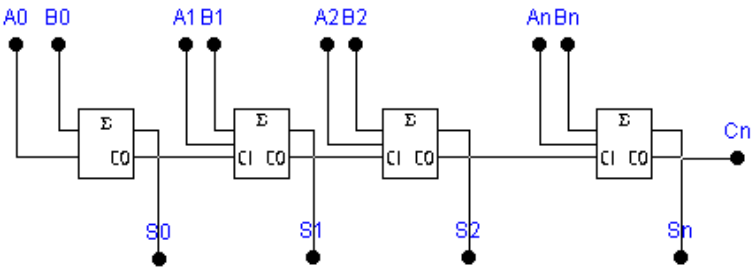
$$= C_{i-1}.(a_i.\, \overline{b_i} + \overline{a_i} \,.b_i) + a_i.\, b_i \,.\, (\overline{C_{i-1}} + C_{i-1}) = \mathbf{C_{i-1}.(a_i \oplus b_i) + a_i.\, b_i}$$

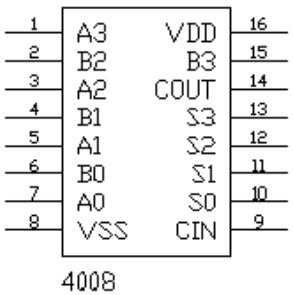
Simbol in funkcijski načrt:





Z združitvijo polovičnega in popolnih seštevalnikov lahko sestavimo n – bitni binarni seštevalnik. Primer 4- bitnega popolnega seštevalnika v IC tehnologiji je 4008.





4008

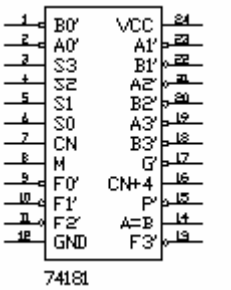
Primeri seštevalnikov v IC tehnologiji:

- 4008 4 – bitni popolni seštevalnik s paralelnim prenosom
- 4038 trojni serijski seštevalnik
- 7480 popolni seštevalnik
- 7482 2 – bitni popolni seštevalnik
- 7483 4 – bitni popolni seštevalnik

Aritmetično funkcijska enota 74181

Active-low Data

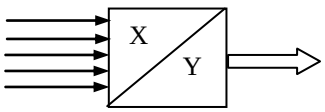
Selection S3 S2 S1 S0	M=H LOGIC FUNCTIONS	M=L; Arithmetic Operations Cn=L (no carry)	Cn=H (with carry)
0 0 0 0	F= $\overline{A}$	F=A MINUS 1	F=A
0 0 0 1	F= $\overline{AB}$	F= AB MINUS 1	F=AB
0 0 1 0	F= $\overline{A}$+B	F= $A\overline{B}$ MINUS 1	F=$A\overline{B}$
0 0 1 1	F= 1	F=MINUS 1(2's comp)	F=Zero
0 1 0 0	F= $\overline{A+B}$	F=A PLUS (A+$\overline{B}$)	F=A PLUS(A+$\overline{B}$) Plus 1
0 1 0 1	F= $\overline{B}$	F=AB PLUS(A+$\overline{B}$)	F=AB PLUS(A+$\overline{B}$) Plus 1
0 1 1 0	F= A "+" B	F=A MINUS B MINUS 1	F= A MINUS B
0 1 1 1	F= A+$\overline{B}$	F= A+$\overline{B}$	F= (A+$\overline{B}$) PLUS 1
1 0 0 0	F= $\overline{AB}$	F= A PLUS (A+B)	F=A PLUS(A+B)PLUS 1
1 0 0 1	F= A "+" B	F= A PLUS B	F=A PLUS B PLUS 1
1 0 1 0	F = B	F= $A\overline{B}$ PLUS (A+B)	F=AB PLUS (A+B)PLUS 1
1 0 1 1	F = A + B	F = (A + B)	F=(A+B)PLUS 1
1 1 0 0	F = 0	F = A PLUS A	F=A PLUS A PLUS 1
1 1 0 1	F = $A\overline{B}$	F= AB PLUS A	F=AB PLUS A PLUS 1
1 1 1 0	F=AB	F=$A\overline{B}$ PLUS A	F=$A\overline{B}$ PLUS A PLUS 1
1 1 1 1	F = A	F = A	F= A PLUS 1



3. 3 Pretvorniki kodov

Kodirnik
je logično vezje, ki vsakemu od 'm' vhodnih signalov priredi 'm' izhodnih kombinacij na 'n' izhodih.

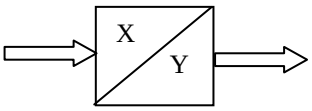
Splošna blokovna shema:



vhodni kanali 'm' izhodni kanali 'n' / velja: $m \leq 2^n$

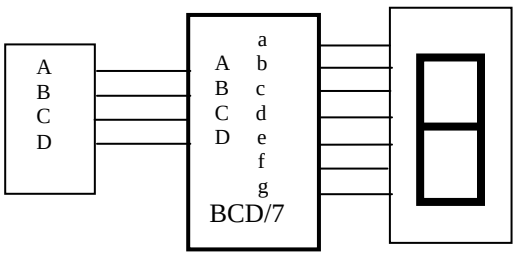
Prekodirnik
je namenjen pretvorbi zapisa iz enega koda v drugi kod, pri čemer je število vhodnih kanalov lahko enako številu izhodnih (prekodiranje med različnimi BCD kodi) ali pa je število vhodnih in izhodnih signalov različno (prekodiranje BCD koda v 7 – segmentni kod: krmiljenje prikazovalnika LED).

Splošna blokovna shema:



vhodni kanali 'm' izhodni kanali 'n' $m = n;\ m \neq n$

Blokovna shema prekodirnika BCD / 7 – segmentni kod za LED prikazovalnik:



Določeni segment LED posveti takrat, ko je na pripadajočem vhodu a, b, c, … logično stanje 0.

Pravilnostna tabela BCD / 7 prekodirnika:

D	C	B	A	a	b	c	d	e	f	g	N
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	0	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	0	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	1	0	0	9
1	0	1	0								x
1	0	1	1								x
1	1	0	0								x
1	1	0	1								x
1	1	1	0								x
1	1	1	1								x

Minimizirane logične enačbe:

$f_a = A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C$

$f_b = A\bar{B}C + \bar{A}BC$

$f_c = \bar{A}B\bar{C}$

$f_d = A\bar{B}\bar{C}\bar{D} + ABC + \bar{A}\bar{B}C$

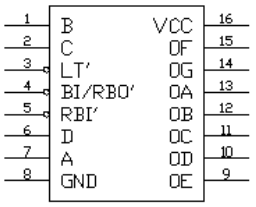
$f_e = A + \bar{B}C$

$f_f = A\bar{C}\bar{D} + B\bar{C}$

$f_g = \bar{B}\bar{C}\bar{D} + ABC$

Primeri IC vezij:

TTL	7446, 7447	BCD / 7-segmentni dekoder - driver
	7448, 7449	BCD / 7-segmentni dekoder
CMOS	4055, 4065	BCD / 7-segmentni dekoder – driver
	4511	BCD / 7-segmentni dekoder
	4555, 4556	2 x 1 / 4 dekoder

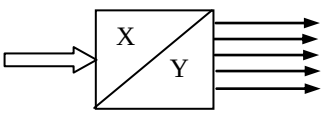


7447

Dekodirnik

je logično vezje, ki pretvori določeni binarni kod iz 'n' vhodov na 'm' izhodnih linij, pri čemer vsaki izhodni liniji pripada enota informacije.

Splošna blokovna shema:



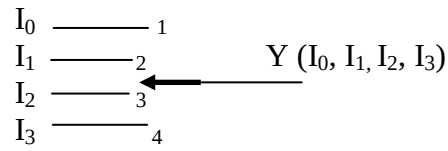
vhodni kanali 'n' izhodni kanali 'm' / velja: $m \leq 2^n$

(BCD / n= 4) (DEC / m = 10)

3. 4 Multiplekserska vezja

Multiplekser – MUX
je element, ki nekemu številu vhodnih signalov priredi odgovarjajoče manjše število izhodnih signalov, ki je odvisno od postavitve krmilnih selektivnih signalov. Velja pravilo, da pri številu 'n' selektivnih signalov lahko pripeljemo na vhod MUX 2^n vhodnih signalov. MUX nam v praksi pretvori množico vhodnih signalov v danem trenutku v en izhodni signal, v naslednjem v drugi izhodni signal itd. Govorimo o časovnem multipleksu.

Stikalni ekvivalent 4/1 MUX:

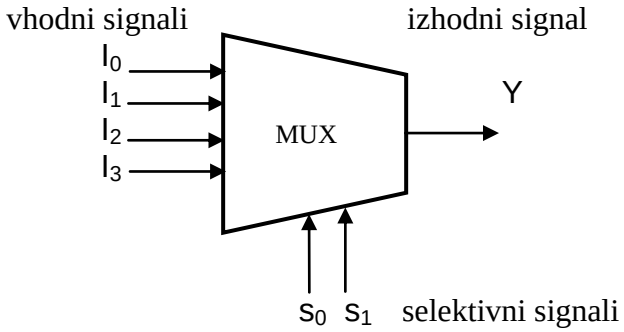


Pravilnostna tabela:

S_1	S_0	I	Y
0	0	I_0	1
0	1	I_1	1
1	0	I_2	1
1	1	I_3	1

Logična enačba: $Y = I_0 .(\overline{S_0} . \overline{S_1}) + I_1 .(S_0 . \overline{S_1}) + I_2 .(\overline{S_0} . S_1) + I_3 .(S_0 . S_1)$

Vezalna shema s simbolom MUX 4/1:

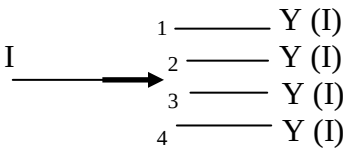


Izvedbe MUX v IC:

74157	4 x 2-kratni 4-polni MUX (2/4)		4051	8/1 MUX
74153	2 x 4-kratni 1-polni MUX (4/1)		4053	3x 3/2 MUX
74352	2 x 4 kratni 1-polni MUX (4/1)		4067	1 – 16 MUX / DMUX

Demultiplekser – DMUX
je element, ki razširi manjše število vhodnih kanalov (lahko je samo en) na večje število izhodnih kanalov. Število izhodnih kanalov je odvisno od števila selektivnih krmilnih signalov in velja: $m \leq 2^n$, pri čemer je 'm' število izhodnih kanalov, 'n' pa število selektivnih vhodov.

Stikalni ekvivalent 1/4 DMUX:



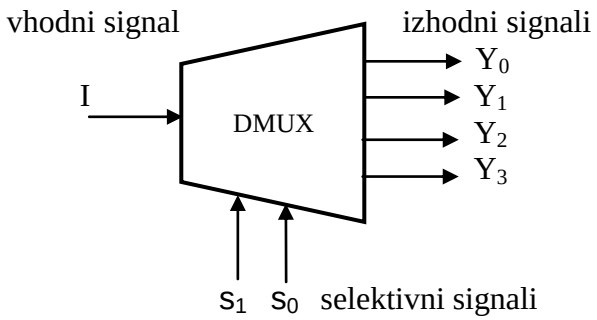
Pravilnostna tabela:

S_1	S_0	I	Y
0	0	I	Y_0
0	1	I	Y_1
1	0	I	Y_2
1	1	I	Y_3

Logične enačbe:

$Y_0 = (\overline{S_1} . \overline{S_0}) . I$
$Y_1 = (\overline{S_1} . S_0) . I$
$Y_2 = (S_1 . \overline{S_0}) . I$
$Y_3 = (S_1 . S_0) . I$

Vezalna shema s simbolom MUX 1/4:

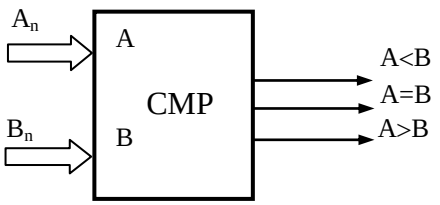


3. 5 Digitalni primerjalnik – komparator CMP

Komparator je logično vezje za primerjanje dveh binarnih števil A in B, kjer je rezultat primerjave lahko:

X:	**A < B**	**Y:**	**A = B**	**Z:**	**A > B**
	x = 1		x = 0		x = 0
	y = 0		y = 1		y = 0
	z = 0		z = 0		z = 1

Blokovna shema:



Pravilnostna tabela za 1-bitni komparator

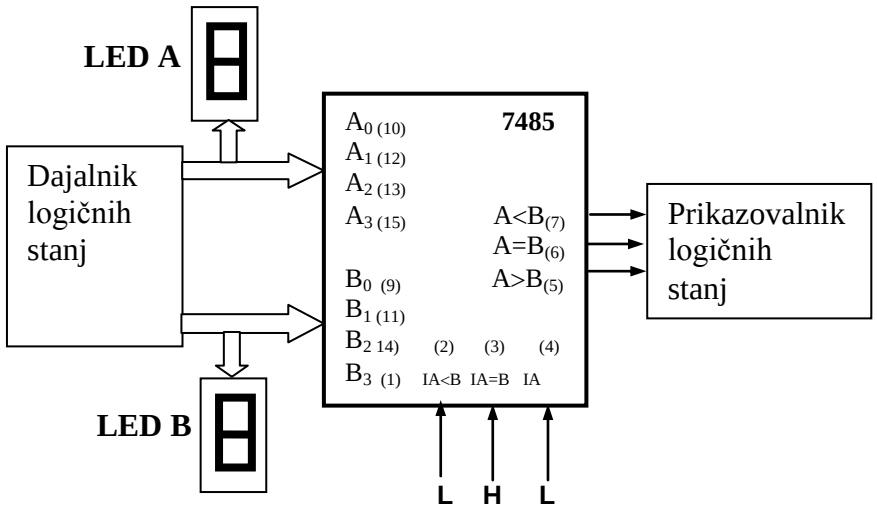
B_0	A_0	x	y	z
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$X = \overline{A}_0 \cdot B_0$

$Y = \overline{A}_0 \cdot \overline{B}_0 + A_0 \cdot B_0 = \overline{A_0 \oplus B_0}$

$Z = A_0 \cdot \overline{B}_0$

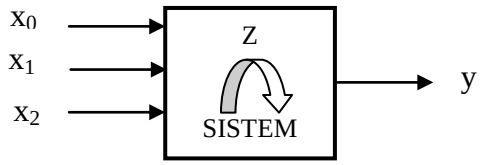
4 – bitni komparator v izvedbi z IC 7485:



4. SEKVENČNA VEZJA

Sekvenčna vezja sestavljajo logična vezja in pomnilni elementi ali pomnilne celice. Izhod iz sekvenčnega vezja 'y' je funkcija dovedenih vhodnih signalov 'x_n' in notranjih stanj 'z_n', v katerem se trenutno sekvenčno vezje nahaja. Sekvenčna vezja delimo na:

- asinhronska; izhodi se postavljajo na osnovi vhodnih in notranjih stanj,
- sinhronska; izhodi se postavljajo na osnovi vhodnih in notranjih stanj ter na osnovi dovedenega takta (clock pulse) in so časovno usklajena.

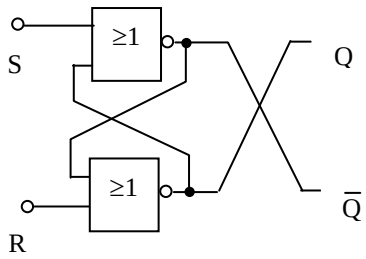


$y = f(z_n, x_n)$
$z_{n+1} = f(z_n, x_n)$ … stanje sistema v trenutku n + 1
z_n … stanje sistema v trenutku n
x_n … stanje vhodnih spremenljivk v trenutku n

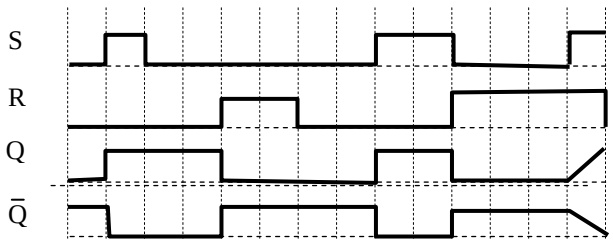
4. 1 Osnovne pomnilne celice

Pomnilne celice (pomnilni elementi, preklopni členi, bistabilni elementi, flip-flopi) so elementi, ki lahko zavzamejo dve stabilni stanji (0,1) in imajo sposobnost, da si ob določenih pogojih svoje stanje zapomnijo.

a) R-S preklopni člen; je osnovna 1 – bitna pomnilna celica



Časovni diagram



Pravilnostna tabela:

R	S	**Q**	**Q'**
0	0	ss	ss
0	1	1	0
1	0	0	1
1	1	/	/

Logične enačbe:

$S = 1 \Rightarrow \overline{Q} = 0;\ R = 0 \Rightarrow Q = 1,\ \overline{Q} = 0$

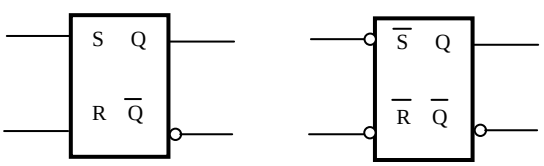
$R = 1 \Rightarrow Q = 0;\ S = 0 \Rightarrow \overline{Q} = 1,\ Q = 0$

ss … staro stanje
/ … nedovoljeno stanje

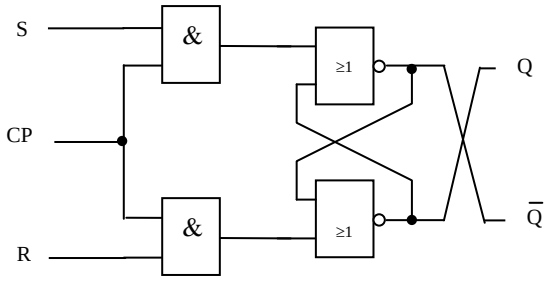
S … vhod za postavljanje (set)
R … vhod za brisanje (reset)

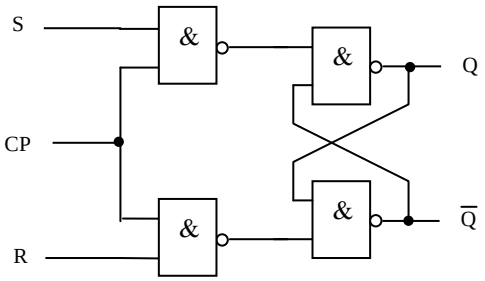
R = 0, S = 0 ⇒ ni zahteve za spremembo stanja
R = 1, S = 1 ⇒ hkratni signal za postavljanje in brisanje – nedovoljeno stanje, v katerem lahko pomnilna celica zavzame katerokoli vrednost: 0 ali 1

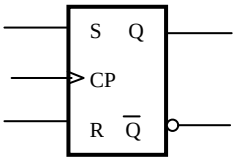
Simbol R-S flip-flopa



Sinhronizirani R-S preklopni člen







Karakteristična tabela pomnilne celice R-S in karakteristična enačba:

Q_n	S_n	R	Q_{n+1}
0	0	0	0 ss
0	0	1	0
0	1	0	1
0	1	1	/
1	0	0	1 ss
1	0	1	0
1	1	0	1
1	1	1	/

$Q_{n+1} = S_n \cdot \overline{R}_n + Q_n \cdot \overline{R}_n = \overline{R}_n \cdot (S_n + Q_n)$ … brez upoštevanja nedovoljenih stanj

$Q_{n+1} = S_n + Q_n \cdot \overline{R}_n$ …. z upoštevanjem nedovoljenih stanj kot redundanc

ss … prenos starega stanja; / … nedoločeno stanje, ki ne sme nastopiti
Q_n … stanje ff v prejšnjem intervalu; Q_{n+1} … stanje ff v sedanjem intervalu

S CP impulzom omogočimo spremembo izhodnega stanja flip-flopa, glede na stanje, ki je trenutno na vhodih R in S. Pri načrtovanjih sekvenčnih vezij je pomembno, da poznamo vhodno kombinacijo R in S v sekvenci n (predhodnem časovnem intervalu), da bomo dobili želeno stanje izhoda v sekvenci n +1 (sedanjem časovnem intervalu) ob znanem stanju Q_n. Postavljanje in brisanje flip-flopa je možno le, če imamo na krmilnem vhodu CP signal 1.

Pravilnostna tabela

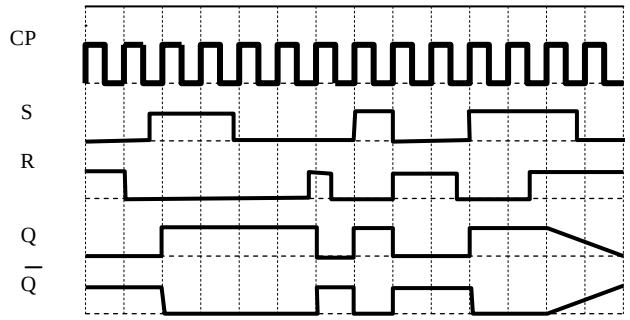
CP	**R**	**S**	Q_{n+1}
1	0	0	Q_n
1	0	1	1
1	1	0	0
1	1	1	/

Vzbujevalna tabela:

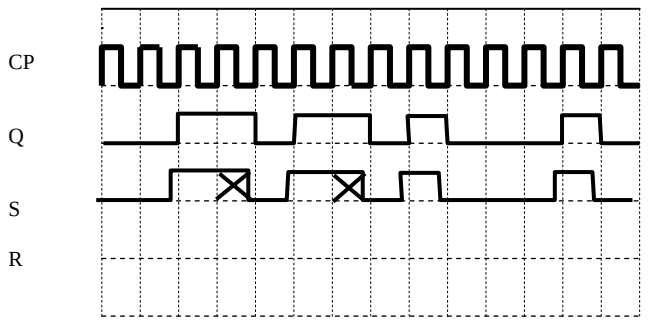
Q_n	Q_{n+1}	S_n	R_n
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

$S_n, R_n = f(Q_n, Q_{n+1})$; X – redundantno stanje: 0,1

Določanje stanja na osnovi danih vhodov



Določanje vhodov S in R na osnovi danega stanja

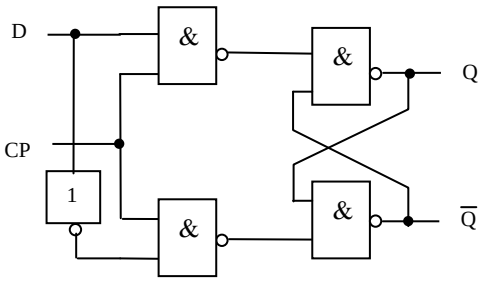


Poleg osnovnega tipa RS celice pa poznamo še celice tipa D, JK in T, ki se razlikujejo po odzivih glede na vhodni signal.

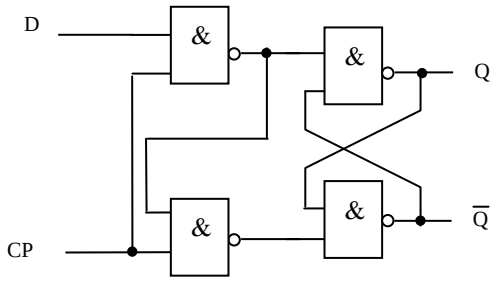
b) D pomnilna celica (Delay)

Kombinacija signalov S = R = 1 privede pri RS preklopnem členu do nepredvidljivega obnašanja celice (lahko se postavi v stanje 0 ali stanje 1). Ta primer bomo izključili na ta način, da na vhod R pripeljemo negiran signal S.

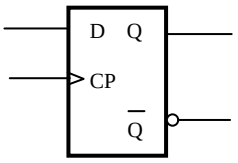
Funkcijski načrt:



Minimizirani D preklopni člen:



Simbol:



Karakteristična tabela pomnilne celice D in karakteristična enačba:

Q_n	D_n	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

$$Q_{n+1} = D_n \cdot \overline{Q}_n + Q_n \cdot D_n = D_n \cdot (\overline{Q}_n + Q_n) = D_n$$

$Q_{n+1} = D_n$ … izhod Q v trenutku n +1 je enak vhodnemu signalu , ki je bil v trenutku n na vhodu D

Stanje na izhodu v trenutku n +1 je enako stanju na vhodu D v trenutku n. To pomeni, da sta stanji vhoda in izhoda časovno zamaknjeni (delay) za en taktni interval. Stikalni preklop povzroči signal 1 na krmilnem vhodu CP. Kakor dolgo traja na vhodu CP signal 1, se lahko stanje flip-flopa spremeni ustrezno vrednosti na vhodu D.

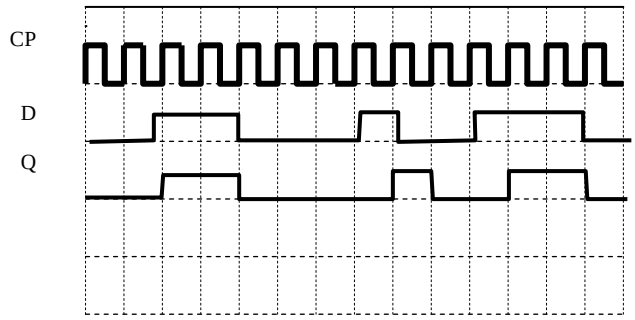
Pravilnostna tabela

D_n	Q_{n+1}
0	0
1	1

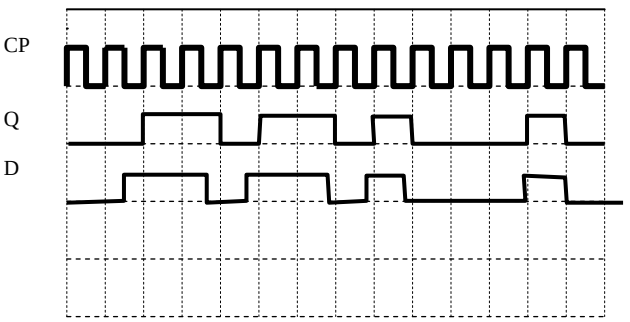
Vzbujevalna tabela: $D_n = f(Q_n, Q_{n+1})$

Q_n	Q_{n+1}	D_n
0	0	0
0	1	1
1	0	0
1	1	1

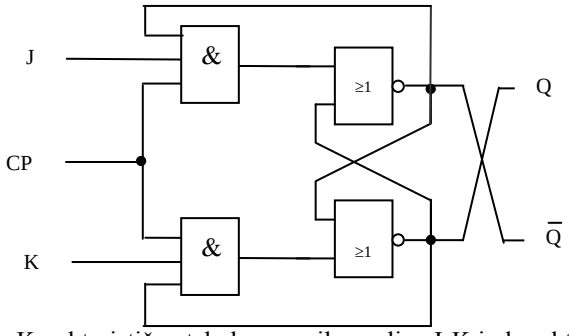
Določanje stanja na osnovi danega vhoda

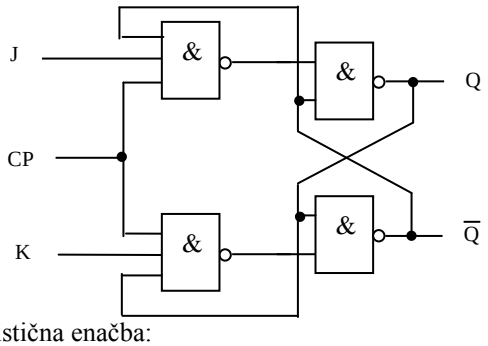


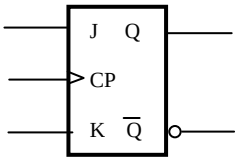
Določanje vhoda na osnovi danega stanja



c) sinhronizirani J – K preklopni člen







Karakteristična tabela pomnilne celice J-K in karakteristična enačba:

Q_n	J_n	K_n	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$$Q_{n+1} = Q_n \cdot \overline{K}_n + \overline{Q}_n \cdot J_n$$

V primeru logičnih enic na krmilnih vhodih J in K se ob signalu na CP izhodno stanje člena prevrže v njegovo negirano stanje. Izhodno stanje se ne spremeni, če sta oba vhoda v stanju 0. Značilnost JK člena je v tem, da sta lahko vhoda hkrati v enakih stanjih, pa bo izhodno stanje vedno definirano.

S CP impulzom omogočimo spremembo izhodnega stanja flip-flopa, glede na stanje, ki je trenutno na vhodih J (SET) in K (RESET). Postavljanje in brisanje flip-flopa je možno le, če imamo na krmilnem vhodu CP signal 1.

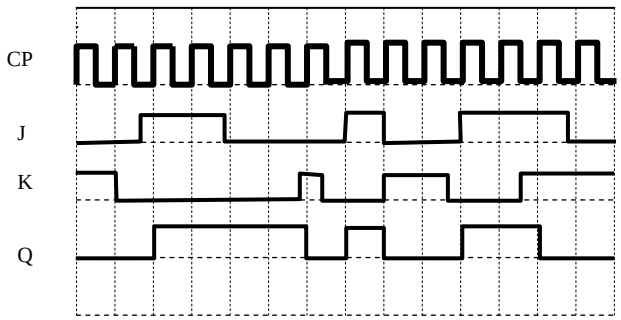
Pravilnostna tabela

CP	**J**	**K**	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q}_n$

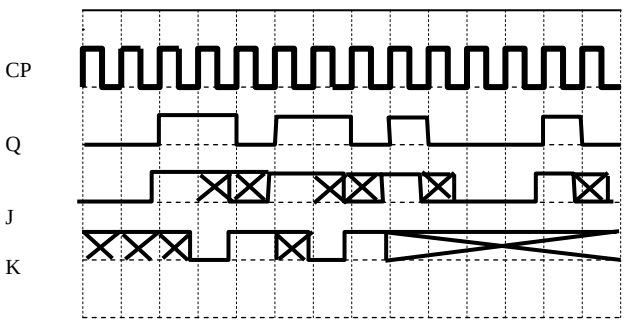
Vzbujevalna tabela: $J_n, K_n = f(Q_n, Q_{n+1})$; X – redundantno stanje: 0,1

Q_n	Q_{n+1}	J_n	K_n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Določanje stanja na osnovi danih vhodov

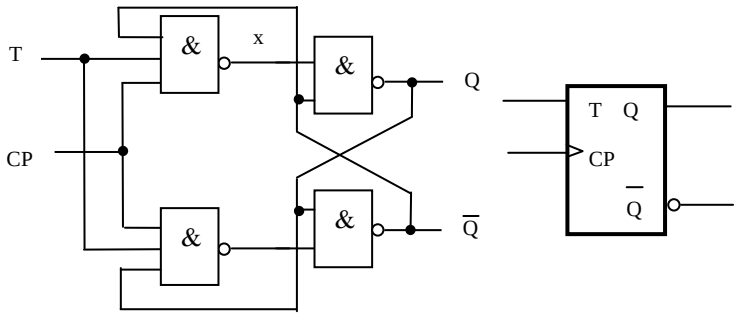


Določanje vhodov J in K na osnovi danega sta



d) T pomnilna celica (trigger flip – flop)

Z združitvijo vhodov J in K na en vhod dobimo T pomnilno celico, katere osnovna značilnost je izmenični preklop izhoda ob negativni fronti CP in postavljenem vhodnem krmilnem signalu T. Pravilnostno tabelo lahko izpeljemo posredno kar iz tabele JK ff.



Karakteristična tabela pomnilne celice T in karakteristična enačba:

Q_n	T_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

$$Q_{n+1} = \overline{Q_n} \cdot T_n + Q_n \cdot \overline{T_n}$$

ali direktno:

	J_n	K_n	Q_{n+1}
→	0	0	Q_n
	0	1	0
	1	0	1
→	1	1	$\overline{Q_n}$

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Pravilnostna tabela

CP	T	Q_{n+1}
1→0	0	Q_n
1→0	1	$\overline{Q_n}$

Proženje T ff

CP	T_n	x
0	0	1
0	1	1
1	0	1
1	1	0

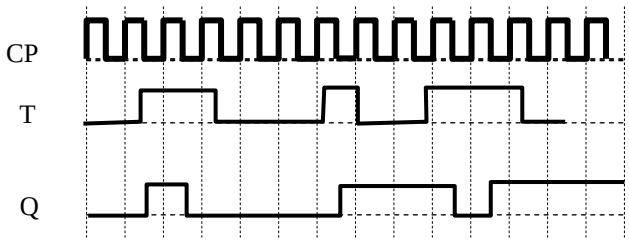
⇒ **x = 0 / T = 1; CP =1** ↓ ⇒ **x = 1 / T =1; CP = 0**

Vzbujevalna tabela:

Q_n	Q_{n+1}	T_n	CP
0	0	0	1→0
0	1	1	1→0
1	0	1	1→0
1	1	0	1→0

Prehod krmilnega signala na vhodu CP iz 1 na 0 (1→0 – zadnja stranica impulza) povzroči spremembo prvotnega stanja na izhodu Q in Q' ob pogoju, da je na vhodu T signal postavljen na 1. Če je na vhodu T signal enak 0, CP impulz ne povzroči spremembe stanja. T flip-flop ob vsaki negativni fronti CP povzroči preklop (T = 1), kar imenujemo dinamično krmiljenje.

Primer:

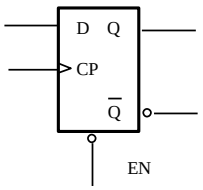


4. 2 Krmiljenje pomnilnih celic

pomeni način vpisa informacije v celice in način hranjenja te informacije. Možnih je več načinov krmiljenja flip-flopov:

- **pomnilne celice z zapahovanjem**; imenujemo jih zapahi ali LATCH-i, uporabljeni so D ff z asinhronskim vhodnim signalom ENABLE; značilna pravilnostna tabela in simbol:

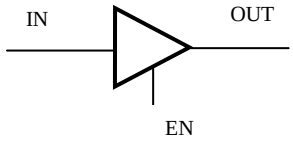
D	EN	Q_{n+1}
0	0	Q_n
0	1	0
1	0	Q_n
1	1	1



Uporabljamo jih za eno ali več bitne zadrževalnike binarnih vrednosti za zadrževanje kratkotrajne informacije.

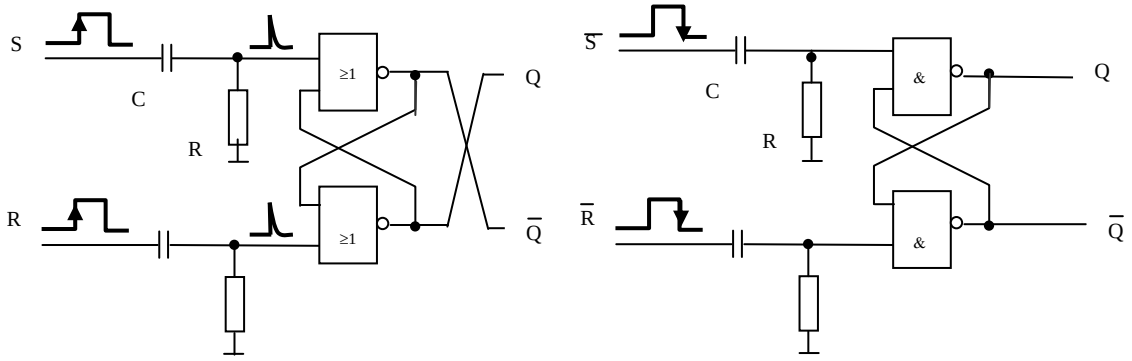
- **pomnilne celice s krmiljenim izhodom**; (3 – state buffer), za to izvedbo uporabimo RS ff, uporabljamo jih za shranjevanje eno bitnih podatkov in niso primerni za števce in pomikalne registre;

EN	IN	OUT
0	0	visoka Z
0	1	visoka Z
1	0	0
1	1	1



- **dinamično krmiljenje pomnilnih celic** je krmiljenje z levim ali desnim bokom impulza (pozitivno ali negativno fronto), ki jo izvedemo z RC diferencirnim vezjem na vhodu; ta način proženja uporabljamo samo v diskretnih bistabilih, v IC izvedbah pa ne; C ≈ 1 nF, R = ≈ 10 kΩ, časovna konstanta RC člena je približno: $T_{RC} \approx 10$ x dvižnega časa (strmine) signala – rise time;

  Primer vezave dinamičnega proženja s pozitivno in negativno fronto:



- **pomnilne celice z asinhronim vpisom**; opremljene so z dodatnimi vhodi SET in CLEAR, preko katerih vpisujemo v celico nove podatke;

- **krmiljenje pomnilnih celic z nizom impulzov**; uporabimo ga pri kaskadnih zaporednih povezavah pri pomikalnih registrih, števcih in delilnikih ter blok povezavah dveh flip-flopov, kjer z enim bokom krmilimo prvi, z drugim bokom pa drugi flip-flop (Master – Slave).

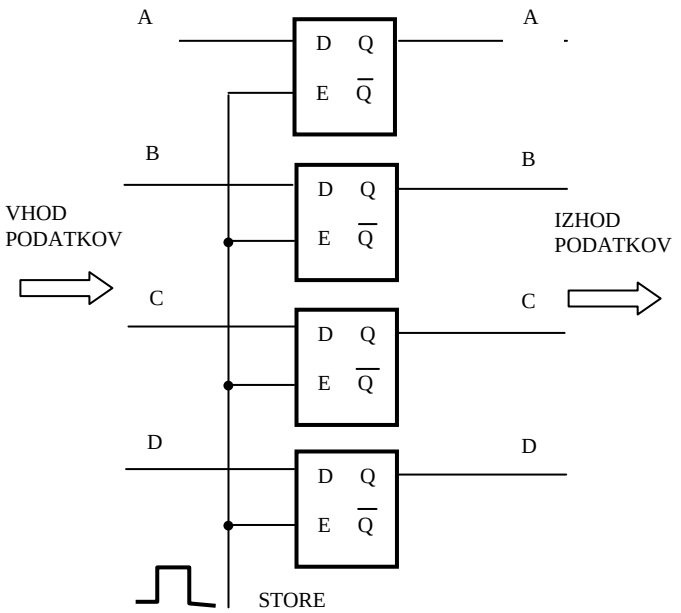
4. 3 Registri

Register je sestavljen iz več eno bitnih pomnilnih celic tipa D, RS ali JK. Vsebina registra je informacija, ki je zapisana v registru. Registre uporabljamo za:
- shranjevanje podatkov; to funkcijo opravljajo pomnilniški blok registri,
- pomikanje vsebine registra; funkcijo opravljajo pomikalni (shift) registri,
- krožno pomikanje informacije, funkcijo opravljajo obročni pomikalni (pomični) registri.

Register – pomnilniški blok

Osnovna celica je D ff oziroma latch. Vhod in izhod podatkov sta izvedena paralelno.

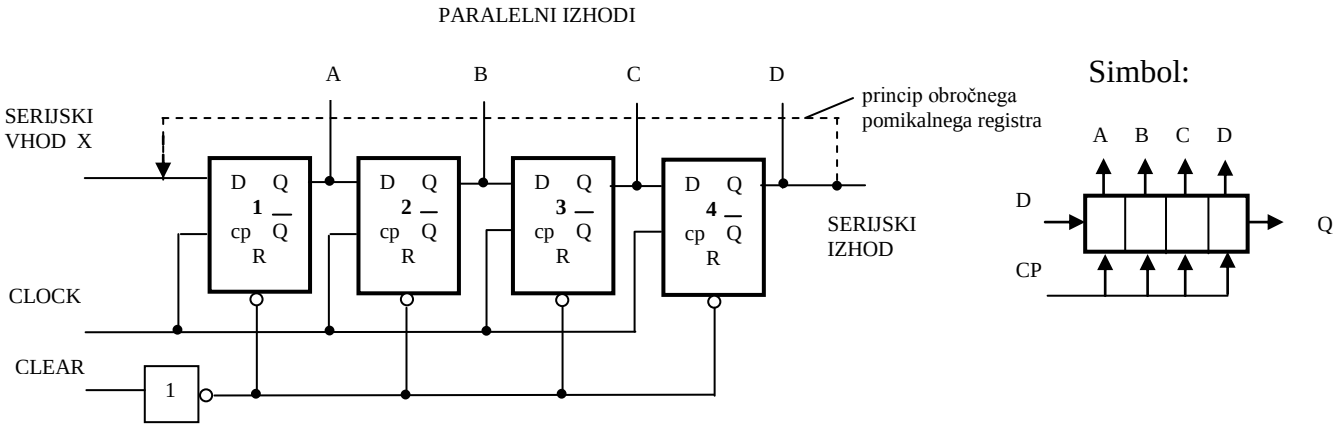
Uporaba:
- shranjevanje podatkov preko paralelnih vmesnikov,
- poljubno (servisno) shranjevanje podatkov,
- sinhronizacija pri paralelnem prenosu podatkov,
- na shemi je prikazan 4- bitni latch register z vpisovalnim krmilnim signalom STORE.



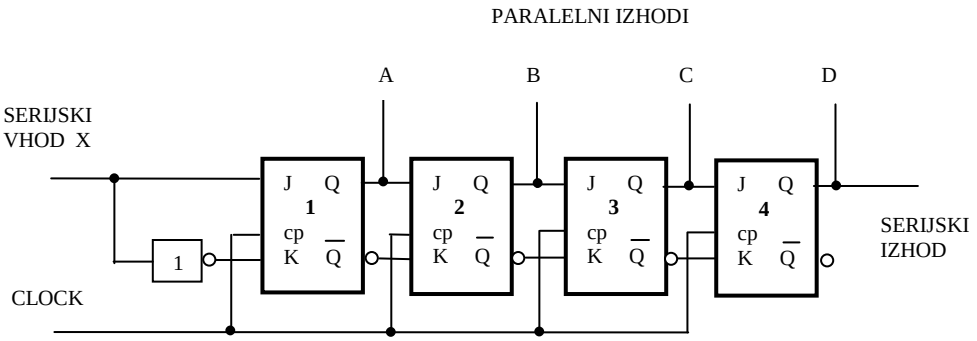
Pomikalni registri - Shift Registers

so verižna vezja, v katerih skupni taktni signal CP pomika v posameznih pomnilnih celicah vsebovano informacijo v celotnem vezju v smeri proti levi ali desni. Dolžina SHIFT registra je odvisna od števila pomnilnih celic, ob vsakem impulzu CP se posamezni biti v registru pomaknejo za eno mesto v desno (shift right), prva pomnilna celica pa sprejme s serijskega vhoda nov bit.

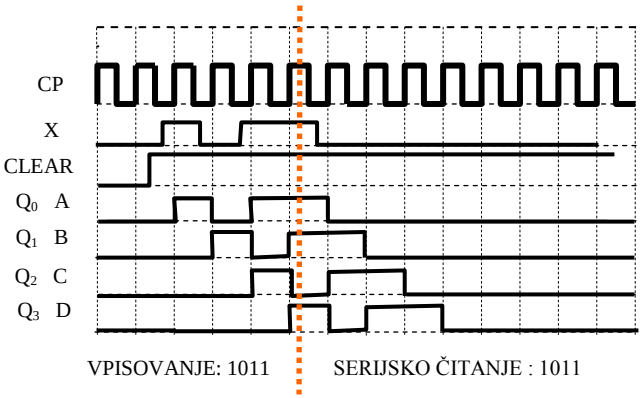
Osnovno vezje 4-bitnega pomikalnega registra tipa D:



Izvedba 4- bitnega pomikalnega registra tipa JK:



Sekvenčni potek pomika bitov v desno:



Tipi pomikalnih registrov IC izvedbi:

7491	8-bitni, serijski, smer – desno
7495	4-bitni, paralelno-serijski, smer – levo/desno
74164	8-bitni, paralelno-serijski, smer – desno
74194	4-bitni, paralelno-serijski, smer – desno/levo
4031	64-bitni, serijski, rotiranje
4035	4-bitni, paralelno-serijski, smer - desno

Tipi registrov po namenu:
- **SISO**: serijski vhod / serijski izhod; uporaba pri sinhronizaciji serijskega prenosa podatkov,
- **SIPO**: serijski vhod / paralelni izhod; uporaba – serijsko/paralelni pretvornik,
- **PISO**: paralelni vhod / serijski izhod; uporaba za paralelno/serijski pretvornik, uporaba asinhronih vhodov PRESET in CLEAR omogoča asinhroni vpis informacij in sinhronski pomik podatkov,
- **PIPO**: paralelni vhod / paralelni izhod; uporaba pri sinhronizaciji paralelnega prenosa podatkov.

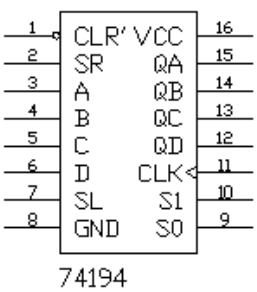
Kriteriji za izbor registrov so predvsem: dolžina registra, tip vhoda in izhoda /vpisa in izpisa, možnost spreminjanja pomika vpisa levo – desno, možnost brisanja vsebine registrov.

74194 – 4 bitni dvosmerni univerzalni pomikalni register

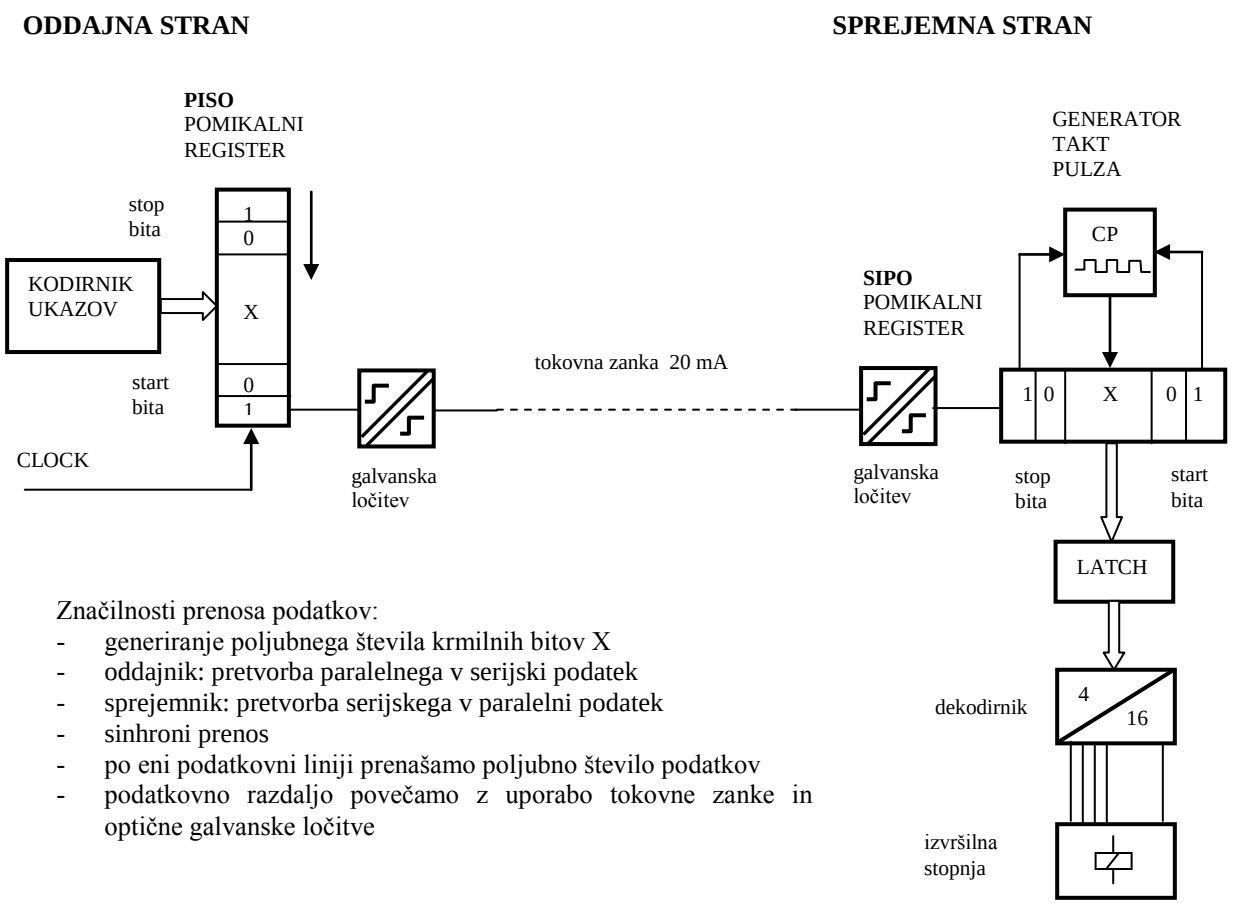
Pravilnostna tabela

	TIP			SERIJSKO		PARALEL				IZHODI			
CLEAR '	S_1	S_0	CLK	LEVO	DESNO	A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	X	X	X	X	X	X	X	X	X	0	0	0	0
1	X	X	0	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
1	1	1	↑	X	X	a	b	c	d	a	b	c	d
1	0	1	↑	X	1	X	X	X	X	1	Q_{An}	Q_{Bn}	Q_{Cn}
1	0	1	↑	X	0	X	X	X	X	0	Q_{An}	Q_{Bn}	Q_{Cn}
1	1	0	↑	1	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	1
1	1	0	↑	0	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	0
1	0	0	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

Priključki



Primer uporabe pomikalnega registra pri prenosu krmilnih signalov – blokovna shema



Značilnosti prenosa podatkov:
- generiranje poljubnega števila krmilnih bitov X
- oddajnik: pretvorba paralelnega v serijski podatek
- sprejemnik: pretvorba serijskega v paralelni podatek
- sinhroni prenos
- po eni podatkovni liniji prenašamo poljubno število podatkov
- podatkovno razdaljo povečamo z uporabo tokovne zanke in optične galvanske ločitve

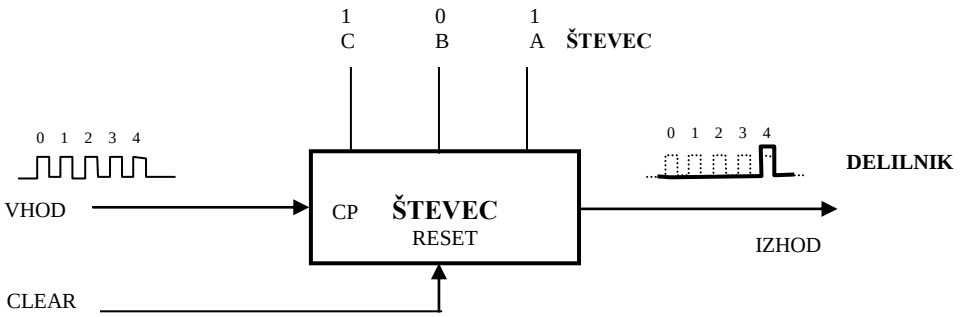
4. 4 Binarni števci in delilniki

Števec je sekvenčni sistem, ki ga lahko uporabimo za:

- štetje ali odštevanje impulzov ter za pomnjenje števila dogodkov v določenem časovnem inrtervalu; govorimo o **števcu – counter**-ju,

- generiranje enega impulza na izhodu po sprejetih N impulzih na svojem vhodu; govorimo o **delilniku**; delilnik deli frekvenco vhodnega signala z N in pri N vhodnih impulzih generira na izhodu en impulz.

Splošna blokovna shema števca

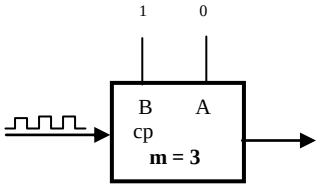


Modul števca 'm' ali SV
je število možnih različnih binarnih stanj, ki jih lahko števec zavzame. Standardne izvedbe:

- dekadni števec z modulom m = 10; uporaba za merilnike frekvence, časovnike in impulzne števce
- šestnajstiški števec z modulom m = 16; 4-bitni delilniki
- števci s poljubnim modulom m = X, pri čemer so poznane izvedbe števcev z JK ff in m = 3 do 32

Števec JK z modulom m = 3

CP	B	A
0	0	0
1	0	1
2	1	0

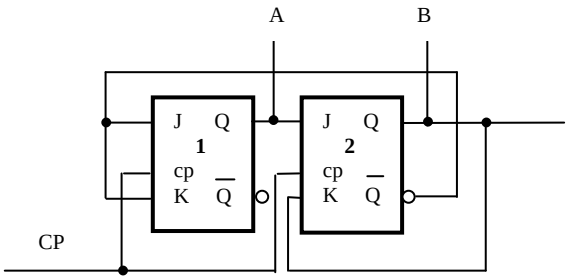


Na osnovi analize delovanja števca in s pomočjo pravilnostne in vzbujevalne tabele JK flip-flopa definiramo logične enačbe oziroma tabelo stanja za vsako pomnilno celico posebej:

ff A: $J_A = B$; $K_A = B$

ff B: $J_B = \overline{A}$; $K_B = \overline{B}$

Vezalna shema števca z modulom m = 3:



Značilnosti vezalne sheme števca:

- vezava obeh ff je sinhronska glede na signal CP
- število ff 'n' ustreza modulu: $m \leq 2^n$
- vhod ff je vezan praviloma direktno ali pa posredno na izhod predhodnega ff, njegov izhod pa praviloma na vhod naslednjega flip-flopa

Logične funkcije vhodov števca za:

m = 5

A: $J_A = \overline{C}$; $K_A = \overline{C}$

B: $J_B = A$; $K_B = A$

C: $J_C = A.B$; $K_C = C$

m = 10

A: $J_A = \overline{A}$; $K_A = A$

B: $J_B = A.\overline{D}$; $K_B = A.\overline{D}$

C: $J_A = A.B$; $K_A = A.B$

D: $J_D = A.B.C$; $K_A = A$

m = 16

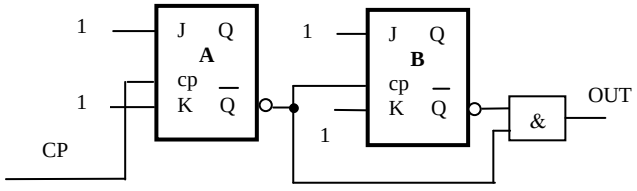
A: $J_A = \overline{A}$; $K_A = A$

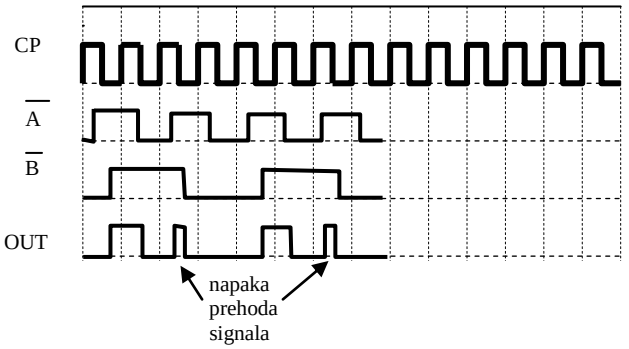
B: $J_B = A$; $K_B = A$

C: $J_A = A.B$; $K_A = A.B$

D: $J_D = A.B.C$; $K_A = A.B.C$

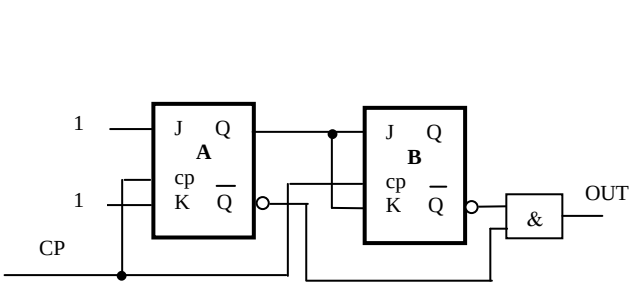
Asinhroni števci se uporabljajo kot 2^n delilniki. Da dobimo 2^n delilnik, moramo izhod predhodnega ff vezati na CP naslednjega ff. Slaba stran teh števcev je prisotnost motilnih signalov kot posledica asinhronih preklopov in časovnih zakasnitev (glitch)

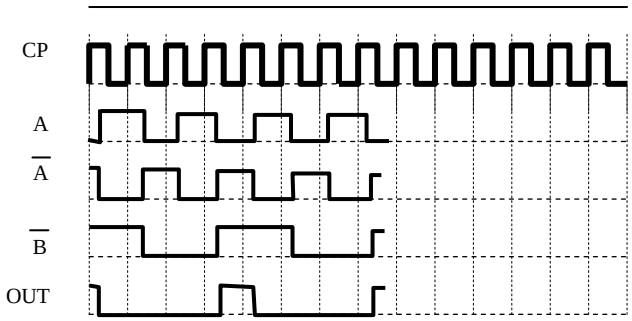




Sinhroni števci

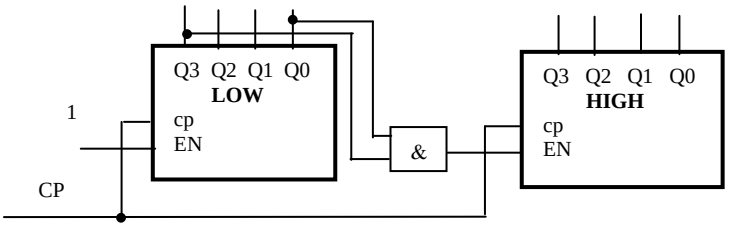
Na krmilne vhode 'cp' flip-flopov pripeljemo vzporedno taktne impulze. Zaradi tega se vsi izhodi preklapljajo sočasno. So hitrejši, ker se časovni zamik pojavlja v enkratni vrednosti. Hitrost delovanja je odvisna od uporabljene tehnologije in veljajo naslednji orientacijski zakasnilni časi: TTL ≈ 10ns, CMOS ≈ 1 –2 μs.





Značilni parametri števcev:

- **nastavljivost** je možnost nastavitve izhodov števca na 0 ali na določeno binarno stanje v okviru modula m (LOAD)
- **kodiranje – dekodiranje**; možnost štetja v binarni kodi, BCD kodi,…
- **krmiljenje**; odvisno je od tipa števca in sicer s pozitivno ali negativno fronto; možna uporaba asinhronih krmilnih signalov (ENABLE: E = 0 ⇒ štetje onemogočeno, E = 1⇒ štetje omogočeno),
- **smer štetja**; poznamo prištevalne, odštevalne in števce GOR / DOL (UP / DOWN)
- **povezljivost** v kaskado; z zaporedno povezavo delilnikov, ki omogočajo deljenje vhodne frekvence z n_1, n_2, n_3, dobimo delilnik s skupnim faktorjem $n = n_1 . n_2 . n_3$; primer kaskadne povezave dveh destiških števcev BCD - 74160 (m = 10), od katerih vsak deli z n= 10, kjer dobimo skupni faktor deljenja n = 100:

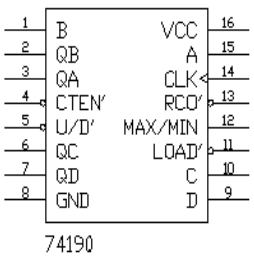


Tipi števcev v IC izvedbi:

7470	m=10, gor, asinhronski, $f_{max.}$ = 32 MHz, ↓
7493	m=16, gor, asinhronski, $f_{max.}$ = 32 MHz, ↓
74160	m=10, gor, sinhronski, $f_{max.}$ = 32 MHz, ↑
74190	m=10, gor/dol, sinhronski, $f_{max.}$ = 30 MHz, ↑
4016	m=10, dol, sinhronski, $f_{max.}$ = 8 MHz, ↑
4018	m=16, dol, sinhronski, $f_{max.}$ = 8 MHz, ↑

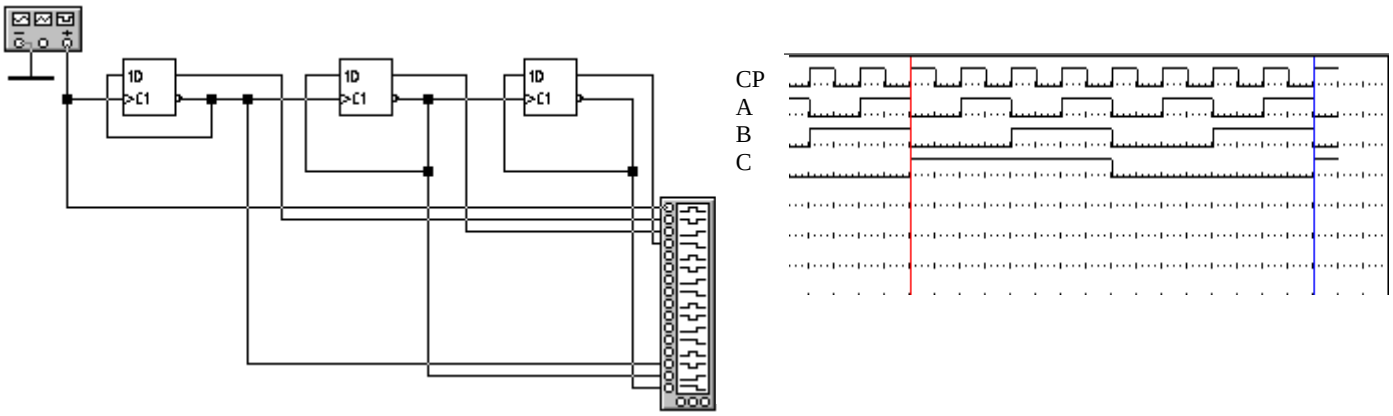
74190 – univerzalni BCD / dekadni nastavljivi števec

CTEN '	D/U'	CLK	LOAD'	A	B	C	D	QA	QB	QC	QD	MAX/MIN	RCO'
0	X	X	0	x	x	x	x	A	B	C	D	1	2
0	1	↑	1	x	x	x	x	štetje dol				1	2
0	0	↑	1	x	x	x	x	štetje gor				1	2
1	X	X	X	x	x	x	x	QA0	QB0	QC0	QD0	1	2



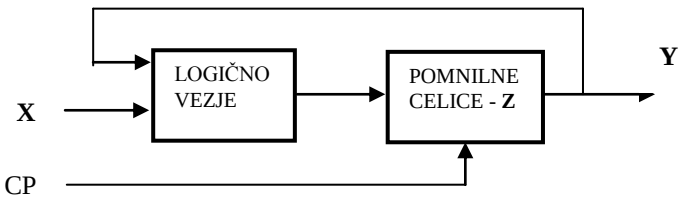
- 1: MAX/MIN = 1 / ŠTEVEC = 9 PRI ŠTETJU GOR IN MAX/MIN = 1 / ŠTEVEC = 0 PRI ŠTETJU DOL
- 2: RCO' = 0 / ŠTEVEC = 9 PRI ŠTETJU GOR IN RCO' = 0 / ŠTEVEC = 0 PRI ŠTETJU DOL

Primer delilnika 2-4-8 v izvedbi z D ff in pripadajoči izhodni signali (izvedba vezja v EWB):



4. 5 Analiza in sinteza sekvenčnih vezij

Sekvenčni sistem sestavljata dve enoti: pomnilne celice (flip-flopi), kot osnovni element sekvenčnih vezij in pripadajoča logična vezja, ki skrbijo za krmiljenje teh pomnilnih celic. Celotni proces se odvija v določenem taktu – CP. Splošna blokovna shema sekvenčnega vezja:



Delitev sinhronih sistemov:

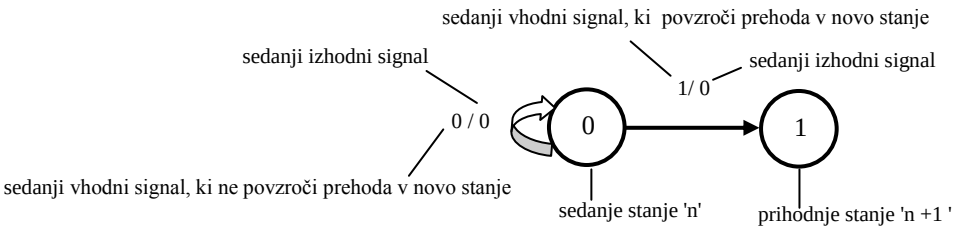
- **Moorov sistem**; sistem, ki nima vhodov x in so izhodna stanja odvisna le od notranjih stanj (števci, …),
- **Mealyev sistem**; sistemi, pri katerih so izhodna stanja odvisna od vhodnih in notranjih stanj; imajo enega ali več vhodov x

Potek analize sekvenčnih vezij:

1. **Tabela stanj**
   tabelarično prikazuje sedanja in naslednja stanja pomnilnih celic pri različnih kombinacijah vhodnih spremenljivk.Vezje z 'm' pomnilnimi celicami lahko zavzame '2^m' stanj, ki jih opišemo z 2^m vrsticami tabele stanj.

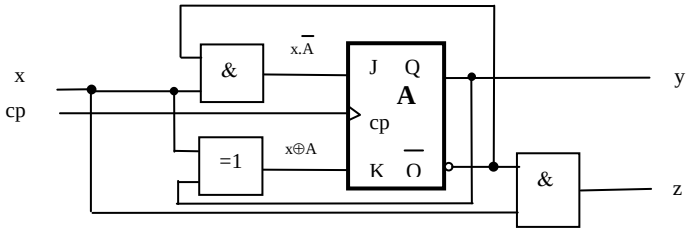
2. **Diagram stanj**
   grafično prikazuje notranja stanja pomnilnih celic in prehode iz enega v drugo stanje ob potrebnih pogojih.

.



3. **Enačbe stanj**
   algebraično opisujejo način, kako posamezne pomnilne celice prehajajo iz sedanjih v naslednja stanja. Leva stran določa naslednje stanje pomnilne celice, desna stran pa preklopno funkcijo, ki opisuje sedanje stanje vhodov in izhodov, pri katerih bo naslednje stanje izhoda 1.
   Enačba ima obliko: $Y_{(n+1)} = f(Y_{(n)} , X_{(n)})$

Primer : Analiza sekvenčnega vezja z 2 izhodoma



Značilnosti vezja:

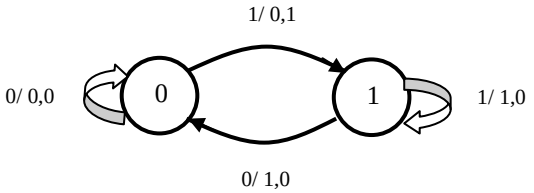
- v vezju je ena pomnilna celica, zato obstajata samo dve stabilni stanji, kar pomeni dve vrstici v tabeli stanj,
- v vezju sta dva izhodna signala, ki ju bomo zapisali v diagramu stanj v zaporedju y,z.

1. Tabela stanj

n	stanja n + 1		izhodi n			
	x = 0	x = 1	x = 0		x = 1	
An	A_{n+1}	A_{n+1}	y_n	z_n	y_n	z_n
0	**0**	**1**	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**	**0**

Enačbi izhodov:
$Y = A$
$Z = X \cdot \overline{A}$

2. Diagram stanj:



3. Enačba stanja (x = 1)

$$A_{n+1} = \overline{A} \cdot X + A \cdot X = X \cdot (\overline{A} + A) = X$$

Enačba stanj kaže, da bo naslednje stanje pomnilne celice le pri stanju 1 na vhodu X in bo neodvisno od predhodnega stanja pomnilne celice.

S pomočjo sinteze izdelamo na osnovi zapisanega algoritma delovanja naprave (diagram stanj, časovni diagram ali opisni algoritem) funkcijski načrt poljubnega sekvenčnega vezja.

Potek sinteze sekvenčnega vezja:

1. opis problema s pomočjo opisa delovanja, časovnih diagramov ali diagrama stanj – izdelava algoritma,

2. sestavljanje tabele stanj,

3. izbor uporabljenih pomnilnih celic,
   JK ff: splošna uporaba
   RS, D: pomikalni registri, binarni števci, delilniki
   T: binarni delilniki

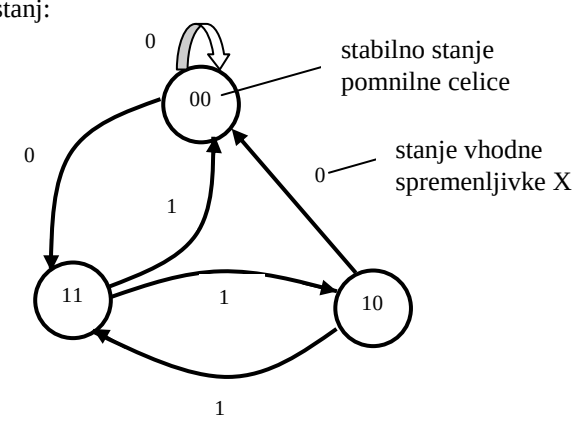
4. sestavljanje vzbujevalne tabele za izbrane pomnilne celice (največkrat JK ff); vzbujevalna tabela sekvenčnega vezja opisuje logično vezje, katerega vhodi so priključeni na izhode pomnilnih celic in vhod X, izhodi logičnega vezja pa so povezani z vhodi posameznih pomnilnih celic,

5. zapis in minimizacija logičnih funkcij,

6. izdelava funkcijskega načrta vezja na osnovi dobljenih logičnih funkcij.

Primer:
Na osnovi podanega diagrama stanj sekvenčnega vezja moramo izdelati ustrezni funkcijski načrt!

1. Diagram stanj:



$m \leq 2^n$

m = 3 - število stabilnih stanj

n = 2 – število potrebnih pomnilnih celic

3. Tabela stanj:

Naslednje stanje pomnilnih celic
Sedanje stanje ff pri sedanjem stanju vhodov

stanje n		$X_n = 0$		$X_n = 1$	
A_n	B_n	A_{n+1}	B_{n+1}	A_{n+1}	B_{n+1}
0	**0**	**0**	**0**	**1**	**1**
0	**1**	**x**	**x**	**x**	**x**
1	**0**	**0**	**0**	**1**	**1**
1	**1**	**0**	**0**	**1**	**0**

3. Izbor pomnilnih celic: JK ff

4. Izdelava vzbujevalne tabele

vhodi logičnega vezja

izhodi log.vezja = vhodi v pomnilne celice

n ff			n+1 ff		vhod v A		vhod v B	
A_n	B_n	X_n	A_{n+1}	B_{n+1}	J_{An}	K_{An}	J_{Bn}	K_{Bn}
0	**0**	**0**	**0**	**0**	**0**	**x**	**0**	**x**
0	**0**	**1**	**1**	**1**	**1**	**x**	**1**	**x**
0	**1**	**0**	**x**	**x**	**x**	**x**	**x**	**x**
0	**1**	**1**	**x**	**x**	**x**	**x**	**x**	**x**
1	**0**	**0**	**0**	**0**	**x**	**1**	**0**	**x**
1	**0**	**1**	**1**	**1**	**x**	**0**	**1**	**x**
1	**1**	**0**	**0**	**0**	**x**	**1**	**x**	**1**
1	**1**	**1**	**1**	**0**	**x**	**0**	**x**	**1**

Vzbujevalna tabela izbranega ff: JK:

Q_n	Q_{n+1}	J_n	K_n
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

5. Funkcije logičnega vezja

A/J_A:

Bn \ An				
	x	x	x	x
	x	x	1	0

Xn

$\mathbf{J_A = X + B_n + A_n}$

B/J_B:

Bn \ An				
	x	x	x	x
		1	1	

Xn

$\mathbf{J_B = X + B_n}$

A/K_A:

Bn \ An				
	1		x	x
	1		x	x

$\mathbf{K_A = \overline{X} + \overline{A_n}}$

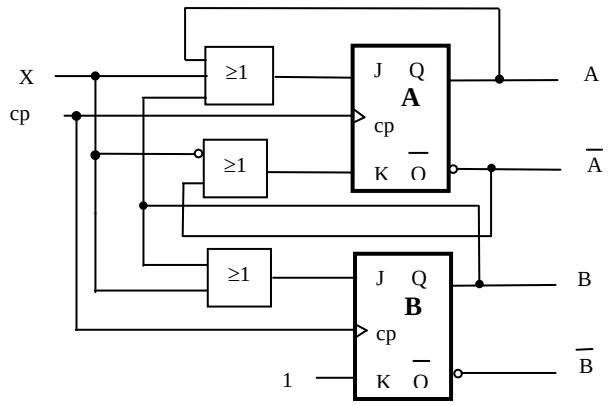
B/K_B:

Bn \ An				
	1	1	x	x
	x	x	x	x

Xn

$\mathbf{K_A = 1}$

6. Funkcijski načrt sekvenčnega vezja



5. POMNILNIKI

Pomnilniki so sistemi za shranjevanje informacij, ki so lahko podatki, operacijski programi, aplikacijski programi ali oboje. Pomnilnike delimo glede na lastnosti:

- **fizikalni princip zapisa:**

  a) magnetni (trdi, mehki disk – uporabni za zunanje pomnilnike)
  b) svetlobni zapis (CD ROM – uporabni za zunanje pomnilnike)
  c) elektronsko vezje (statični / ff, dinamični / kondenzator v mikrovezju , programirljive matrike – uporabni za notranje in delovne pomnilnike ter registre v mikro-procesorjih)

- **način dostopa do lokacije:**

  a) naključni (dostopni čas do katerekoli lokacije je enako dolg za vse podatke, uporabni so kot delovni pomnilniki RAM (Random-Access-Memory), ROM (Read-Only-Memory), PROM (Programmable-Read-Only-Memory)
  b) zaporedni (dostopni časi so različni, princip tračnih enot)

c) krožni (pomnilni medij je oblikovan v kolobar)
d) direktni (kombinacija pomnilnikov z zaporednim in krožnim dostopom, uporaba pri diskovnih in disketnih pomnilnikih).

Karakteristični parametri pomnilnikov so:

- **funkcija pomnilnika** določa namen in uporabnost vrste pomnilnika. V splošnem jih delimo na:

  a) zunanji pomnilniki (eksterni) – hard disk, CD ROM, diskete, tračne enote, kasete; njihova značilnost je velika kapaciteta in manjša hitrost dostopa,

  b) notranji (interni) polprevodniški pomnilniki

  - bralno pisalni pomnilniki (RAM); delimo jih na statične (zapis informacij po bitih v pomnilne celice – uporaba pri večjih hitrostih - registri) in dinamične (zapis informacije v oblik električnega naboja v kondenzator mikrovezja – uporaba za delovne pomnilnike z večjo kapaciteto)

  - bralni pomnilniki:
  - **neprogramirljivi ROM** pomnilniki (samo za branje)
  - **programirljivi PROM** (enkratni tovarniški zapis), PLA
  - **reprogramirljivi** – **EPROM** (Erasable-Read-Only-Memory), **EAROM** (Erasable-Alterable-Read-Only-Memory), **EEPROM**(Electrical-Erasable-Read-Only-Memory), GAL

- **kapaciteta pomnilnika,** ki jo določa število pomnilniških lokacij, do katerih imamo bralni ali pisalni dostop. Lokacije so običajno eno ali osem bitne – byte. Osnovne enote za merjenje kapacitete:
  1 kb (kB) - 2^{10} = 1024 bitov (byte-ov) ; primer: 16 kb = 2^{14} = 16 384 bitov
  1 Mb (MB) - 2^{10} kb = 2^{20} bitov = 1.048.576 bitov
  1 Gb (GB) - 2^{10} Mb = 2^{30} bitov = 1,0737 . 10^9 bitov

- **dostopni čas pomnilnika** je čas, ki preteče od trenutka naslavljanja lokacije do trenutka, ko je podatek na razpolago. Pomnilniki z naključnim dostopom imajo krajše čase kot pomnilniki z direktnim dostopom. Glede na hitrost razdelimo pomnilnike po naslednji hierarhiji:

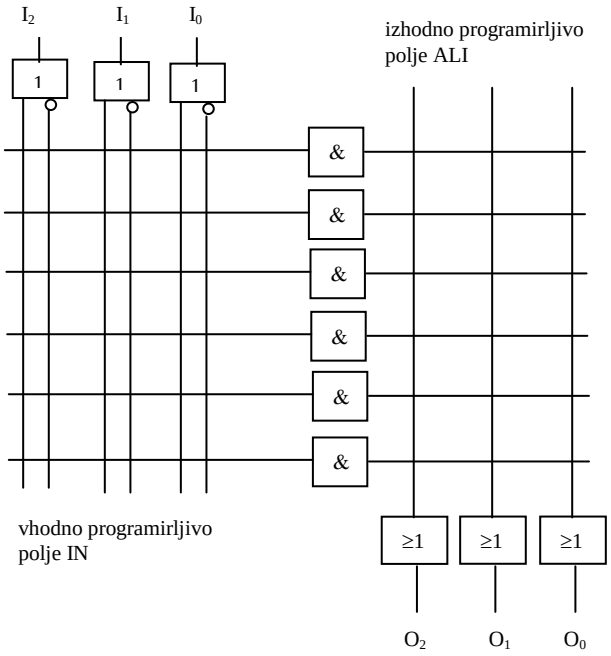
  1. registri CPU (najhitrejši)
  2. predpomnilnik v CPU (CACHE)
  3. predpomnilnik na osnovni plošči (ROM)
  4. delovni pomnilnik RAM, PROM, EPROM
  5. zunanji sekundarni pomnilnik (hard disk, CD ROM)

5. 1 Programirljiva logična polja - PLA

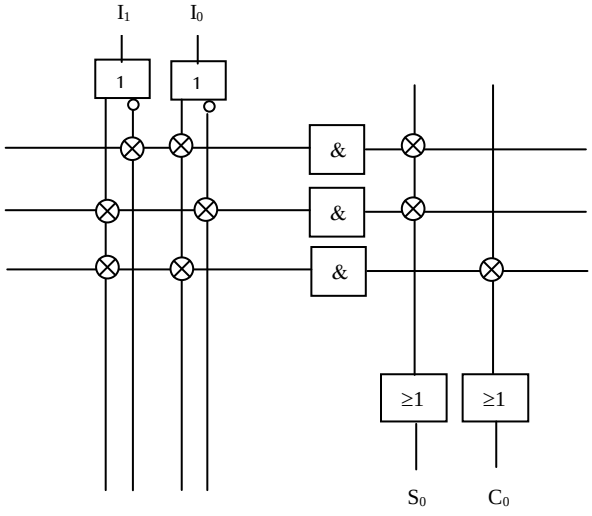
Programirljiva logična polja ali matrike (PLA – Programmable Logic Array) so programirljivi sistemi, s katerimi lahko realiziramo logične ali aritmetične funkcije. Uporabimo jih pri sistemih z večjim številom vhodnih spremenljivk, ki so logično povezane z izhodnimi kanali. PLA uporabljamo za kodiranje in de-kodiranje, za zamenjavo diskretnih logičnih elementov, mikroprogramiranje itd. Logična mreža PLA je običajno že vgrajena v mikroprocesorje, njena naloga je dekodiranje ukazov in kontrola posameznih enot. Funkcijo PLA lahko zamenja pomnilnik tipa ROM.

PLA sestavlja vhodna IN mreža in izhodna ALI mreža, katerih presečne točke (varovalke) lahko programiramo (prežgemo) in tako realiziramo poljubno logično funkcijo. Presečne točke v mreži pomenijo posamezni zapis vhodnih in izhodnih spremenljivk. Tipi logičnih polj so:
- PROM, programirano vhodno polje IN in programirljivo izhodno polje ALI,
- PLA, programirljivo polje IN in programirljivo polje ALI (univerzalni tip),
- PAL, programirljivo polje IN in programirano polje ALI.

Splošna struktura PLA:



Primer programiranega PLA za polovični seštevalnik:



$S_0 = I_0 \oplus I_1$
$C_0 = I_0 \cdot I_1$

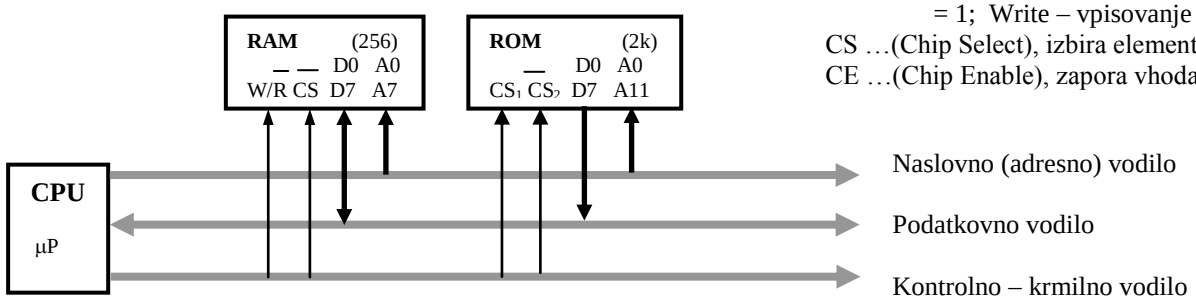
GAL integrirana vezja
so v osnovi vezja tipa PLA s tem, da imajo v svoji strukturi pomnilne celice (D ff) in so kompleksnejša. Osnovne značilnosti:

- povezave so izvedene v obliki MOS FET-ov
- na izhodu imajo programirljivo makro – celico
- omogočajo zaščito pred branjem oziroma kopiranjem
- omogočajo večkratno reprogramiranje
- omogočajo nadzorovan in krmiljen izhod, ki je lahko tudi neposredna funkcija vhodnega signala
- GAL vezja programiramo s pomočjo programatorja, ki ga priključimo na paralelni vmesnik PC
- program je sestavljen iz zaglavja (tekst), deklaracijskega bloka (opis integriranega vezja, priključni pini in popis vhodnih in izhodnih kanalov) in funkcijskega bloka (zapis ustreznih logičnih enačb)

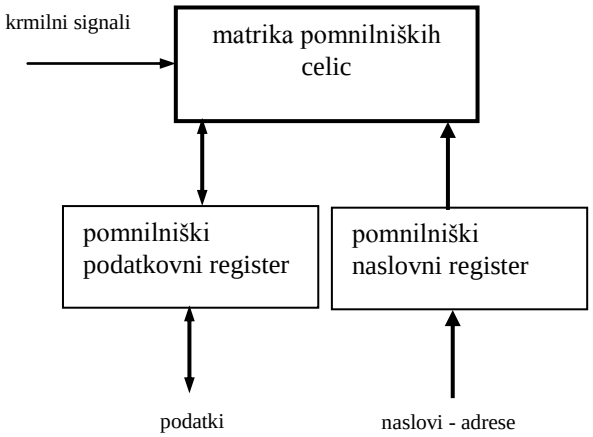
5. 2 Splošna organizacija pomnilnikov

Povezava pomnilnikov na sistemsko vodilo

D_0-D_n …n-bitna podatkovna linija
A_0-A_m …m-bitna naslovna linija
W/R' … = 0; Read – branje
= 1; Write – vpisovanje
CS …(Chip Select), izbira elementa
CE …(Chip Enable), zapora vhoda



Splošna blokovna shema bralno-pisalnega pomnilnika (RAM) z naključnim dostopom

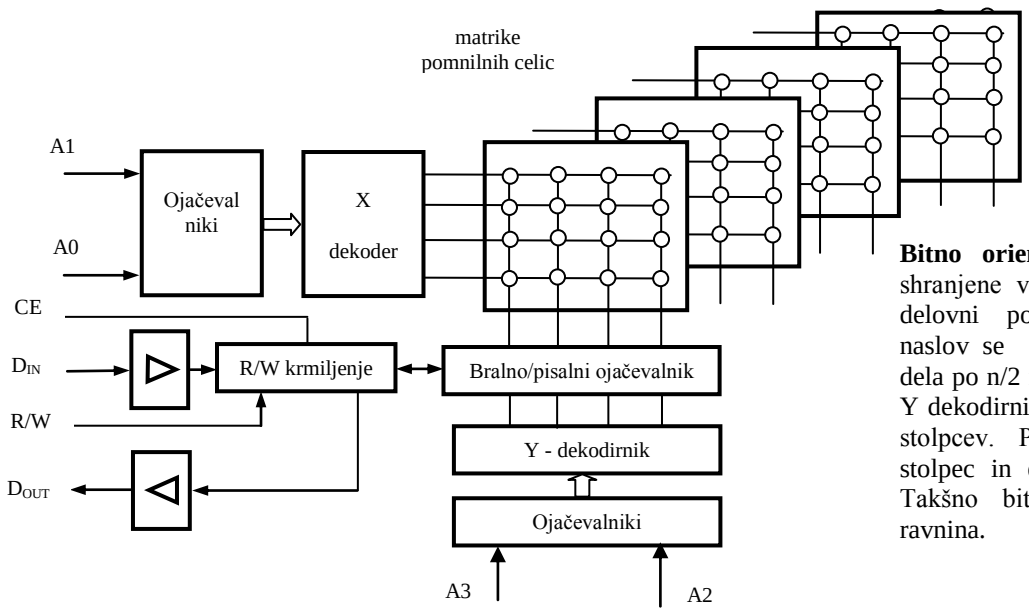


Pomnilniški pisalni cikel: pri vpisu podatka v pomnilnik vpiše procesor najprej naslov pomnilniške lokacije v pomnilniški naslovni register, nato s krmilnim signalom za vpis vpiše podatek v naslovljeno lokacijo.

Pomnilniški bralni cikel: pri branju podatka iz pomnilnika se najprej vpiše naslov lokacije podatka v naslovni register, nato pa se generira krmilni signal za branje, ki povzroči izpis naslovljenega podatka v podatkovni register.

Organizacija pomnilnika z naključnim dostopom

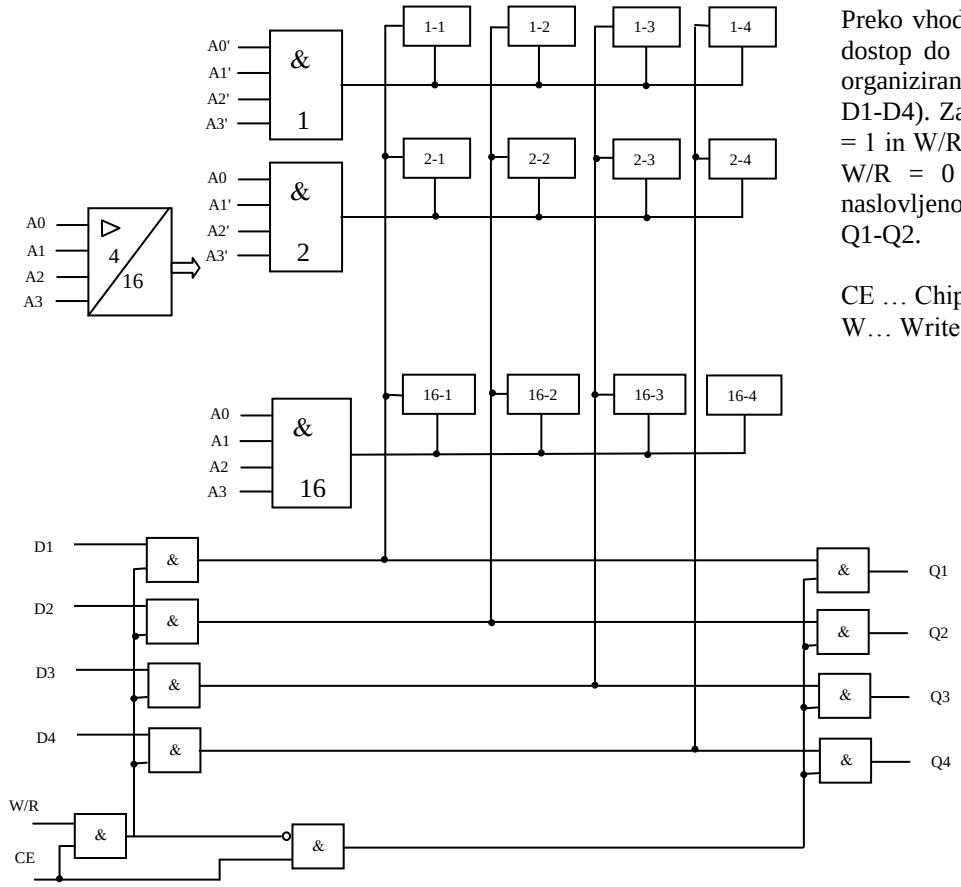
Pomnilne celice so vezane v matriko, ki jo sestavlja določeno število vrstic in stolpcev. Presečišča vrstic in stolpcev so posamezne pomnilne celice. Glede na dostop do pomnilnih celic ločimo bitno in besedno orientirane pomnilnike.



Bitno orientirani pomnilnik; besede so shranjene v več bitnih ravninah, (DRAM-delovni pomnilnik). n-bitni pomnilniški naslov se preko krmilnika razdeli na dva dela po n/2 naslovnih bitov, ki se preko X in Y dekodirnika dekodirajo v $2^{n/2}$ vrstic in $2^{n/2}$ stolpcev. Pomnilniški naslov določa en stolpec in ena vrstica, ki sta v stanju 1. Takšno bitno matriko imenujemo bitna ravnina.

Besedno orientirani pomnilniki

Pomnilne celice so razporejene druga poleg druge tako, da vrsta predstavlja eno pomnilniško besedo (8, 16 bitov). Teh besed je 2^n, če imamo v naslovu 'n' bitov; (ROM, SRAM).

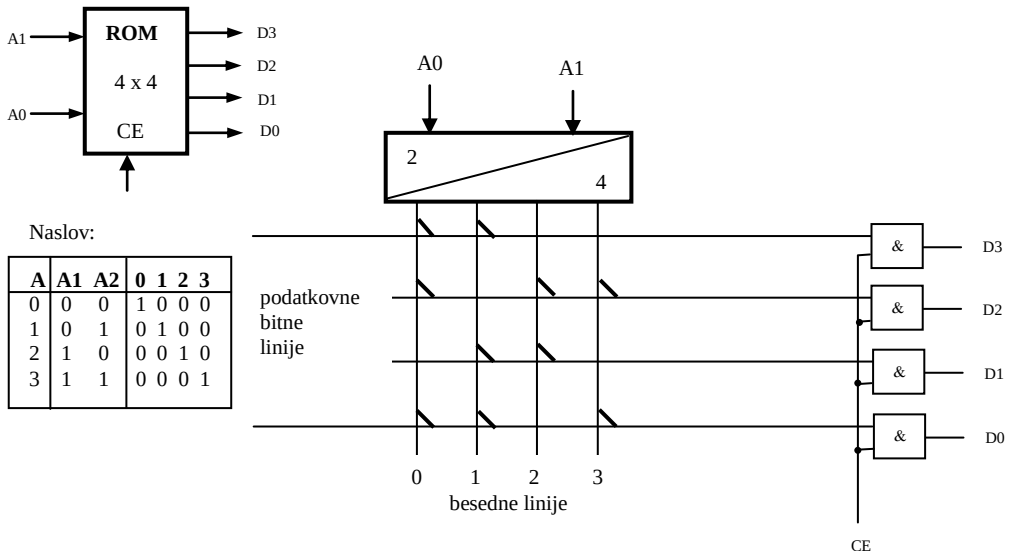


Preko vhodov z naslovi A0-A3 dosežemo dostop do posamezne vrstice po besedah organiziranega pomnilnika (4-bitne besede D1-D4). Za vpis ene besede mora biti CE = 1 in W/R = 1. Če je na vhodu CE = 1 in W/R = 0 lahko preko vhodov A0-A3 naslovljeno besedo preberemo na izhodih Q1-Q2.

CE … Chip Enable – zapora vhoda
W… Write; R … Read

5. 3 Bralni pomnilniki – ROM

Njihovo vsebino lahko samo beremo, v pomnilnik vpisana vsebina pa se ohrani tudi po izklopu napajalne napetosti. Primer organizacije 4-bitnega ROM-a:



Naslov:

A	A1	A2	0	1	2	3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

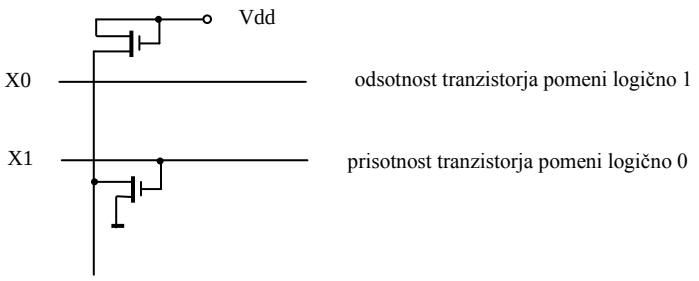
Podatek:

A	D0	D1	D2	D3
0	1	0	1	1
1	1	1	0	1
2	0	1	1	0
3	1	0	1	0

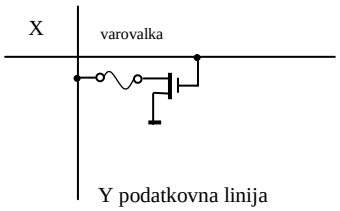
Osnovne lastnosti bralnih pomnilnikov:

- uporaba za vpis zagonskih programov, generatorji znakov, v alfanumeričnih prikazovalnikih,…
- organizacija pomnilnika je besedna (običajno 8 –bitna), kar pomeni, da je celotni program zapisan v enem IC
- izhodne linije dekodirnika so vezane z izhodnimi linijami podatkov in predstavljajo posamezne besedne linije,
- pomnilne celice so sestavljene iz programirljivih povezav in veznih elementov (diode, tranzistorji),
- izhodi iz matrike so podatkovne linije za posamezne podatkovne bite D_n
- kapaciteta ROM: 32 kB, 64 kB do 1 MB, dostopni časi so 100 do 300 ns.

- ROM pomnilniška celica:



- PROM pomnilniška celica:

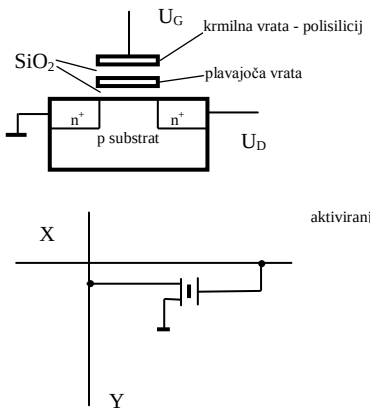


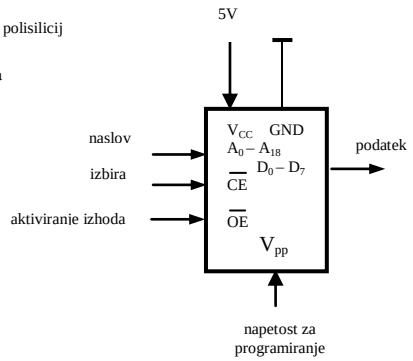
PROM pomnilniki

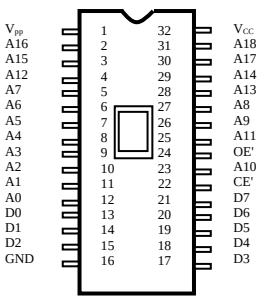
- vsebino pomnilnika programiramo s programirno napravo, pri čemer tvorimo diodno tokovno povezavo med vodniki vrstic in stolpcev; možno je enkratno programiranje,
- bitna celica je tranzistor ali dioda, ki je preko električno oslabljene povezave - varovalke (fusible link) povezana v matriko,
- zaradi nezmožnosti ponovnega programiranja uporabljamo te pomnilnike vse manj.

EPROM pomnilniki

- kot podatki se hranijo električni naboji, vpisani v pomnilno celico MOSFET tranzistorja s plavajočimi vrati (FAMOS),
- vpis podatkov v pomnilnik opravimo preko programirne naprave in po posebnem algoritmu; programirne napetosti V_{pp} so običajno 12,5V in 21V,
- obstojnost električnega naboja na plavajočih vratih in s tem podatka je velika (cca.10 let), podatke lahko izbrišemo z UV svetlobo visoke intenzitete , ki prodira v mikrovezje preko odprtine (okence) na IC,
- orientacija pomnilnika je besedna z dolžinami besede 8 in 16 bitov, kapacitete so od 2k besed pa do 1 M besede, dostopni časi so med 100 in 300 ns,
- EPROM pomnilniška celica: Shema EPROM 512 k x 8: EPROM 27C040 v DIP ohišju:







EEPROM (E^2ROM) pomnilniki

- je električno programiran pomnilnik z možnostjo brisanja, ki ga dosežemo s povišano napetostjo, ki povzroči razelektritev vrat,
- imajo tanjšo izolacijsko plast okoli plavajočih vrat (cca. 10 nm),
- kapaciteta pomnilnikov je do 64 kb,
- čas dostopa je reda μs, pisanja in brisanja pa reda ms.

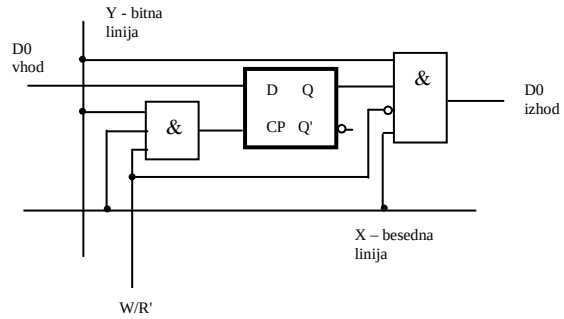
5. 4. Bralno-pisalni pomnilniki RAM

Pomnilnike tipa RAM uporabljamo v računalniških in krmilnih sistemih kot delovne pomnilnike. Vanje začasno shranjujemo dele operacijskega sistema, stalne in trenutne programe in vmesne rezultate. Ko jim odvzamemo napajalno napetost, izgubijo vpisano vsebino. V primerih, ko teh podatkov ne smemo izgubiti, opremimo RAM pomnilnike z rezervnim napajalnim virom (aku-baterije). Podatke lahko večkrat na novo vpišemo in jih tudi brišemo, pri čemer se struktura pomnilnih celic ne poškoduje. Pomnilnike RAM ločimo glede na izvedbo pomnilnih celic na statične pomnilnike SRAM in dinamične DRAM.

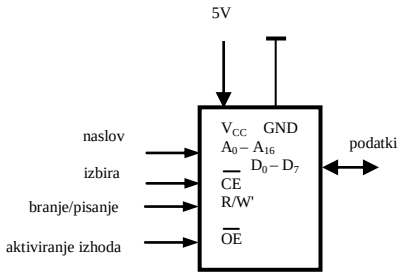
a) SRAM pomnilniki

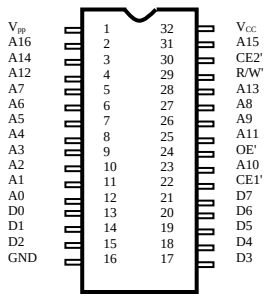
Osnovne značilnosti so
- pomnilna celica je zgrajena kot flip-flop, kapaciteta je zaradi tega manjša, cena na bit pa večja,
- podatek ohranjajo tako dolgo, dokler je prisotna napajalna napetost,
- imajo večjo energijsko porabo in trajno obremenjujejo napajalnik,
- kratki dostopni časi (izvedba MOS – reda 10 ns, bipolarna izvedba – reda 1 ns).
- organizacija pomnilnika je lahko bitna (uporabljenih 6 – 8 tranzistorjev) ali besedna (uporabljeni 4 tranzistorji).

Organizacija pomnilne celice SRAM:



Shema SRAM 128 k x 8 (TC551001):





Priključki SRAM:
- A_0 do A_n naslovni priključki (n = 7 – 16),
- D_0 do D_m podatkovni priključki (m = 3,7,15 za besedno orientacijo, pri bitno orientiranih SRAM, pa je samo en podatkovni priključek),
- CE, CS vhod za izbiro ali selekcijo IC (Chip Enable, Chip Select),
- $\overline{OE}$ vhod za sprostitev izhodov (Output Enable),

R/$\overline{W}$ ali WE krmilni vhod za postavitev bralnega oz. pisalnega cikla; W je vedno negiran, tako je vpis definiran z nizkim stanjem, branje pa z visokim.

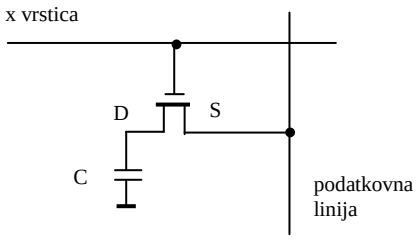
Krmilni signali statičnega RAM:

	CS	R/W	OE	D7 – D0
vpis	0	0	1	D7 – D0
branje	0	1	0	podatki za vpis
neaktivno	1	X	1	stanje visoke Z

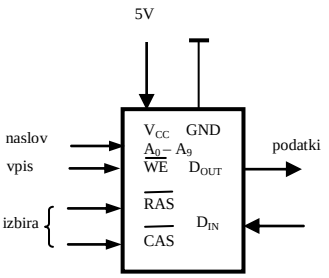
b) DRAM pomnilniki

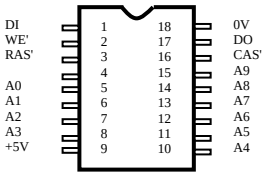
Osnovne značilnosti:
- pomnilna celica je zgrajenega iz tranzistorja in integriranega kondenzatorja, v katerega zapisujemo podatek; kapaciteta teh pomnilnikov je zaradi enostavnosti vezja večja od pomnilnikov tipa SRAM
- dielektrik pri pomnilni celici je zelo slab, naboj je majhen, zato je potrebno sprotno osveževanje vsebine kondenzatorja (refresh), kar zahteva dodatno vezje,
- energijska raba je manjša in manj obremenjujejo napajalnik,
- zahtevajo daljše dostopne čase (reda 100 ns), kar zahteva sistemske rešitve: predpomnilniki, pomnilniške banke,
- praviloma so bitno organizirani in pokrivajo področje večjih delovnih pomnilnikov.

Organizacija pomnilne celice DRAM:



Shema dinamičnega RAM 1M x 1:





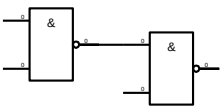
Priključki in signali dinamičnega RAM 1M x 1:

- RAS (Row Addres Strobe); omogoči vpis naslova vrstice v vrstični dekodirnik
- CAS (Column Addres Strobe); omogoči vpis naslova stolpca v stolpični dekodirnik
- A_0 do A_9 omogoča bitna organizacija pomnilnika; naslov se deli na polovico (vrstica in stolpec), kar zahteva dodatna naslovna dekodirnika za stolpce in vrstice,
- krmilnik pomnilnika DRAM deli naslov na dve polovici, od tega je ena polovica namenjena za stolpce, druga polovica pa za vrstice (naslovno multipleksiranje) in generira časovno premaknjena krmilna signala RAS in CAS,
- DI (Data Input) in DO (Data Output) sta značilna podatkovna kanala za bitno organizacijo pomnilnikov,
- WE (Write Enable) nadomešča krmilni signal R/W.

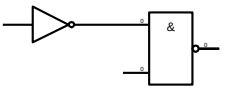
6. Vezja za prilagajanje logičnih nivojev

- faktor FAN OUT (faktor izhodne obremenljivosti)
- FAN OUT je visok, če je na TTL priključen MOS vhod
- MOS vezje lahko poganja en TTL (štiri LS TTL)
- kompatibilnost MOS – TTL

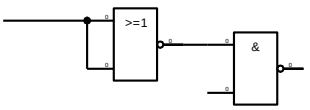
- napajanje C-MOS je 5V, priključitev C-MOS na TTL:



CMOS/5V LSTTL/5V
en vhod

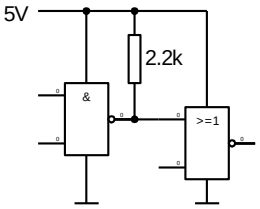


MOS BUFFER TTL - katerikoli
4049 - invertiran
4050 - neinvertiran

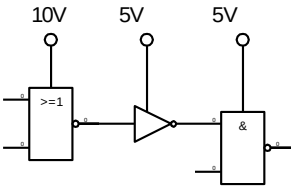


CMOS/5V standardni TTL
4001
4002

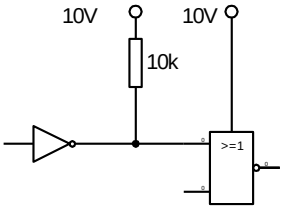
- priključitev TTL na C-MOS: **- napajanje C-MOS 10V:**



TTL C-MOS

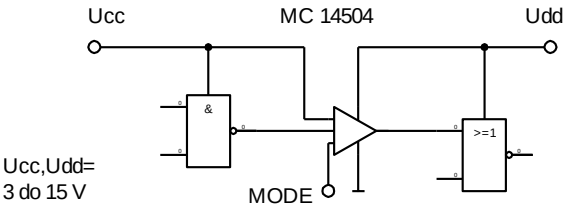


CMOS DRIVER TTL
CMOS
4049,4050

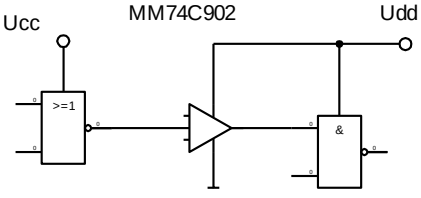


TTL DRIVER CMOS
7406, 7406
74016, 7417
OC izhod

- posebna prilagoditvena vezja:



CMOS ali TTL Input: TTL CMOS
Output:CMOS



CMOS Ucc > Udd CMOS ali TTL

- **izhodna vezja** (driverji, posebni elementi z I = 250 mA /74C908, 74C918, uporaba tranzistorjev)