LOGIČNE STRUKTURE IN SISTEMI I

prof. dr. Andrej Dobnikar

5. februar 2004

Splošne informacije

- Predavatelj: prof. dr. Andrej Dobnikar
  govorilne ure: četrtek, 13:00 - 14:00
  kabinet v 8. nadstropju
- Asistent: dr. Uroš Lotrič
  govorilne ure: torek, 10:00 - 11:00
  Laboratorij za adaptivne sisteme in paralelno procesiranje v 8. nadstropju
- Predavanja in vaje:
  - 3 ure predavanj na teden
  - 3 ure vaj na teden (1 ura avditornih vaj in 2 uri laboratorijskih vaj)
    Obvezna prisotnost na vajah 80% (10/13)
- Cilji predmeta:
  - spoznavanje osnov preklopne logike in preklopnih funkcij,
  - snovanje odločitvenih in sekvenčnih preklopnih vezij,
  - povezovanje z osnovami strojne opreme
  - pregled osnovnih elementov digitalnih naprav.
- Literatura:
  - Jernej Virant: Logično snovanje računalniških struktur in sistemov, Založba FE, Ljubljana, 1987.
  - R. H. Katz: Contemporary Logic Design, The Benjamin/Cummings Publishing Company, 1994.
  - A. D. Friedman, P. R. Menon: Theory&Design of Switching Circuits, Digital System Design Series, 1975.
  - Mira Trebar, Osnove logičnih vezij, Založba Fe in Fri, Ljubljana, 2002.

Kazalo

Poglavje 1

Zgodovinski pregled

Današnja preklopna ali Boolova algebra, ki predstavlja matematični okvir računalniških znanosti, ima izvor v algebri razredov v okviru teorije množic, v logiki izjav in v predikatnem računu.

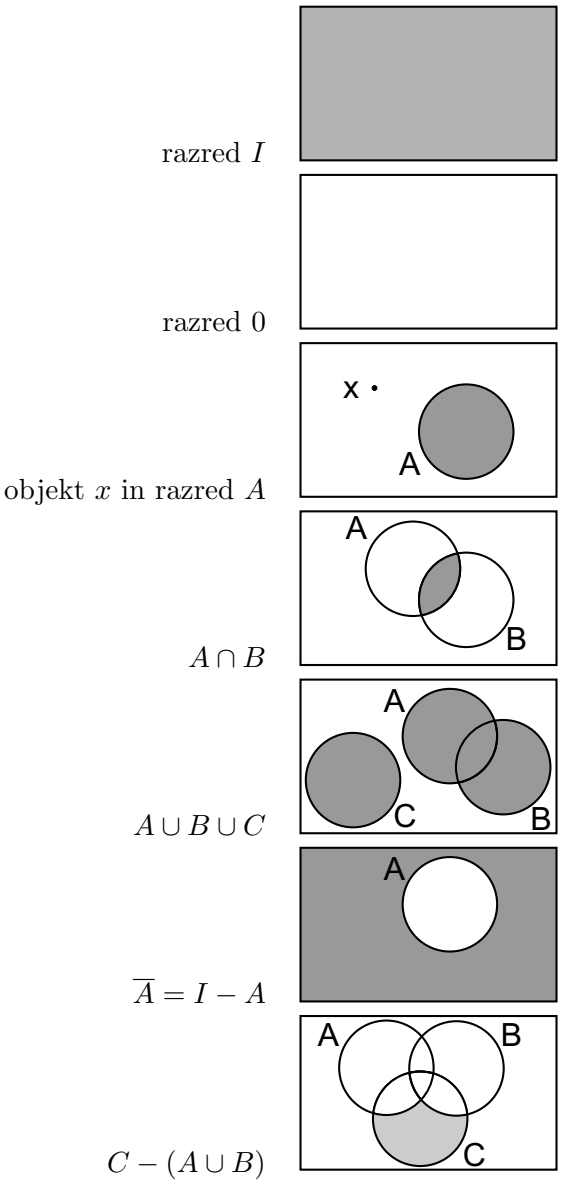
1.1 Algebra razredov

- Razred je skupina objektov, ki imajo isto lastnost. Pripadnost objekta x razredu ali množici X podaja izraz

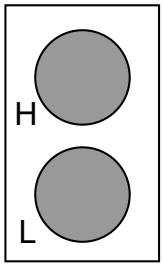
  $$x \in X$$

- Razred $\emptyset$ je prazen razred, ki nima nobenega elementa.
- Razred I je poln razred, ki ima vse elemente, zato ga imenujemo tudi univerzalni razred.
- Razredi so med seboj v relacijah. Na primer, razred X je vsebovan v razredu Y, $X \subset Y$.
- Nove razrede lahko dobimo s pomočjo operacij nad razredi. Najbolj znane operacije so:
  - presek: $X \cap Y$,
  - unija: $X \cup Y$,
  - komplement: $\overline{X} = I - X$ in
  - razlika: $X - Y$.
- Velja:
  - $X \cup \overline{X} = I$ in
  - $X \cap \overline{X} = \emptyset$.

- Razrede podajamo grafično s pomočjo Vennovih diagramov:



- V računalništvu pogosto uporabljamo dve množici napetosti. Prvo označujemo z L (iz angleške besede Low, nizko), kamor sodijo napetosti med 0 in 1.5 V, drugo pa s H (iz angleške besede High, visoko), kjer so napetosti med 3.5 V in 5 V.



I: napetosti med 0 in 5V
H: napetosti od 3.5 V do 5 V
L: napetosti od 0 V do 1.5 V

1.2 Logične izjave

- Logične izjave opisujejo določene pripadnosti, in so lahko pravilne (1, *ang.* true) ali pa nepravilne (0, *ang.* false).

  Primer 1:
  Izjava X: Napetost je visoka (H).
  Če je izjava pravilna, je $X = 1$, sicer je $X = 0$.

- Pomembnejše operacije nad izjavami so
  - disjunkcija izjav (logični ali),
  - konjunkcija izjav (logični in) in
  - negacija izjav.

1.3 Logika predikatov

- Predikatno polje $P = \{a, b, c, ...\}$, je množica objektov, nad katerimi testiramo določeno lastnost oziroma predikat. Na primer,

$$p(x) = \begin{cases} 1, & \text{če lastnost } p \text{ ustreza objektu } x \\ 0, & \text{sicer} \end{cases}$$

- Predikate povezujemo s kvantorji. Najbolj znani so naslednji:
  - kolektivnost: $(\forall x)p(x)$,
  - obstojnost: $(\exists x)p(x)$ in
  - stroga obstojnost $(\exists! x)p(x)$.

- Veljajo naslednje relacije:
  - $\overline{(\forall x)p(x)} = (\exists x)\overline{p(x)}$
  - $\overline{(\exists x)p(x)} = (\forall x)\overline{p(x)}$

  Primer 2:
  $P = \{x_1, x_2, ...\}$ = vsi prisotni v predavalnici
  $p(x) \equiv$ pozna Boolovo algebro
  $(\forall x)p(x) = 0$: vsi prisotni ne poznajo Boolove algebre,
  $(\exists x)p(x) = 1$: predavatelj in nekateri študentje so seznanjeni z Boolovo algebro,
  $(\exists! x)p(x) = 1$: predavatelj je seznanjen z Boolovo algebro.

Poglavje 2

Boolova algebra

2.1 Definicija Boolove algebre

Boolova ali preklopna algebra je definirana z algebraičnim sistemom

$$\langle X, \vee, \&, P1 - P6 \rangle \ ,$$

kjer je

- $X = \{x_1, x_2, x_3, \ldots\}$ množica preklopnih spremenljivk, ki zavzamejo le dve vrednosti – 0 in 1,
- $\vee$ disjunkcija (supremalni operator, logični ali),

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

- $\&, \cdot$ konjunkcija (infimalni operator, logični in),

x_1	x_2	$x_1 \& x_2$	$(x_1 \cdot x_2)$
0	0	0	
0	1	0	
1	0	0	
1	1	1	

- $P1 - P6$ pa so postulati Boolove algebre:
  - P1: zaprtje: disjunkcija ($\vee$) in konjunkcija (&) zavzameta le vrednosti 0 in 1, tako kot vhodne spremenljivke:

    $$x_1 \vee x_2 \in X$$
    $$x_1 \cdot x_2 \in X$$

  - P2: obstajata aditivna (0) in multiplikativna (1) konstanta, tako da velja:

    $$x_i \vee 0 = x_i$$
    $$x_i \cdot 1 = x_i$$

– P3: velja komutativnost disjunkcije in konjunkcije

$$x_1 \vee x_2 = x_2 \vee x_1$$
$$x_1 \cdot x_2 = x_2 \cdot x_1$$

– P4: velja distributivnost disjunkcije in konjunkcije

$$x_1 \vee (x_2 \cdot x_3) = (x_1 \vee x_2) \cdot (x_1 \vee x_3)$$
$$x_1 \cdot (x_2 \vee x_3) = (x_1 \cdot x_2) \vee (x_1 \cdot x_3)$$

Pozor: v običajni algebri operatorja seštevanja $(+)$ in množenja $(\cdot)$ nista enakovredna, in zato

$$a + (b \cdot c) \neq (a + b) \cdot (a + c)$$
$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

– P5: obstaja komplement $\overline{x}_i$ (negacija) tako, da velja

$$x_i \vee \overline{x}_i = 1$$
$$x_i \cdot \overline{x}_i = 0$$

Pozor: takšna oblika negacije v običajni algebri ne obstaja.

– P6: Eksistenca
Za veljavnost Boolove algebre sta potrebni vsaj dve spremenljivki iz X, za kateri velja:

$$\min_X(x_i, x_j) : x_i \neq x_j$$

2.2 Teoremi

Teoremi ali izreki Boolove algebre so logične enačbe, ki morajo biti dokazane. Dokazujemo jih lahko s postulati, s pravilnostnimi tabelami ali s kontaktnimi shemami.

2.2.1 Pomembnejši teoremi

- Teoremi z eno spremenljivko
  - $x \vee 1 = 1$
    $x \cdot 0 = 0$
  - $x \vee x = x$
    $x \cdot x = x$ (teorema idempotence)
  - $\overline{\overline{x}} = x$
- Teoremi z dvema spremenljivkama
  - $x \vee x \cdot y = x$
    $x \cdot (x \vee y) = x$ (teorema absorpcije)
  - $(x \vee \overline{y}) \cdot y = x \cdot y$
    $(x \cdot \overline{y}) \vee y = x \vee y$

  - $(x \vee y) \vee \overline{x} = 1$
    $(\overline{x} \cdot \overline{y}) \cdot x = 0$
  - $\overline{x \vee y} = \overline{x} \cdot \overline{y}$
    $\overline{x \cdot y} = \overline{x} \vee \overline{y}$ (de Morganova teorema)

- Teoremi s tremi spremenljivkami
  - $(x \vee y) \cdot (y \vee z) \cdot (z \vee \overline{x}) = (x \vee y) \cdot (z \vee \overline{x})$
    $x \cdot y \vee y \cdot z \vee z \cdot \overline{x} = x \cdot y \vee z \cdot \overline{x}$
  - $x \vee (y \vee z) = (x \vee y) \vee z = x \vee y \vee z$
    $(x \cdot y) \cdot z = x \cdot (y \cdot z) = x \cdot y \cdot z$ (teorema asociativnosti)
    Pozor: Asociativnost omogoča razširitev operacije iz dveh na tri ali več spremenljivk ob ohranitvi pomena operacije. Disjunkcija in konjunkcija, na primer, sta supremalna oziroma infimalna operacija ne glede na število vhodnih spremenljivk.

2.2.2 Dokazovanje teoremov

Primer 3: Dokaz teorema $x_1 \vee \overline{x}_1 \cdot x_2 = x_1 \vee x_2$

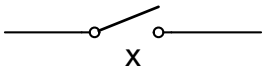
- s postulati:

$$x_1 \vee \overline{x}_1 \cdot x_2 \overset{P4}{=} (x_1 \vee \overline{x}_1) \cdot (x_1 \vee x_2) \overset{P5}{=} 1 \cdot (x_1 \vee x_2) \overset{P2}{=} x_1 \vee x_2$$

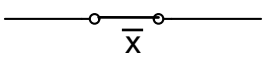
- s pravilnostno tabelo:

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

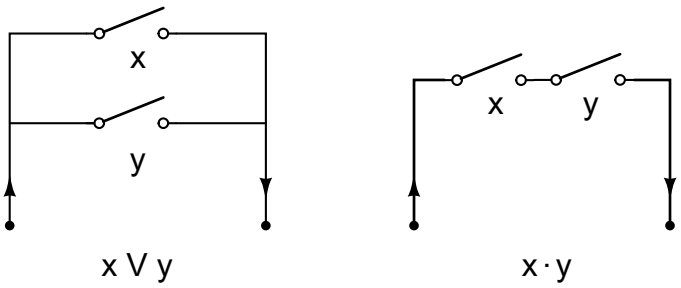
- s stikalno (kontaktno) shemo:
  - spremenljivka x



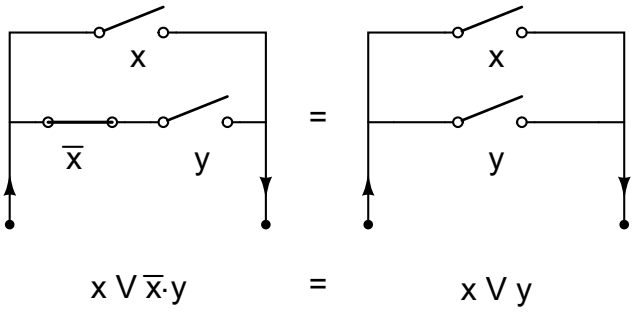
    Ko je $x = 0$ je stikalo odprto (razklenjeno), ko je $x = 1$ pa je stikalo zaprto (sklenjeno).
  - spremenljivka $\overline{x}$



    Ko je $x = 0$ je stikalo zaprto (sklenjeno), ko je $x = 1$ pa je stikalo odprto (razklenjeno).
  - disjunkcijo rišemo kot paralelno (vzporedno) vezavo stikal, konjunkcijo pa kot serijsko (zaporedno) vezavo stikal



Dokaz:



Primer 4: Dokaz teorema $x \vee 1 = 1$:

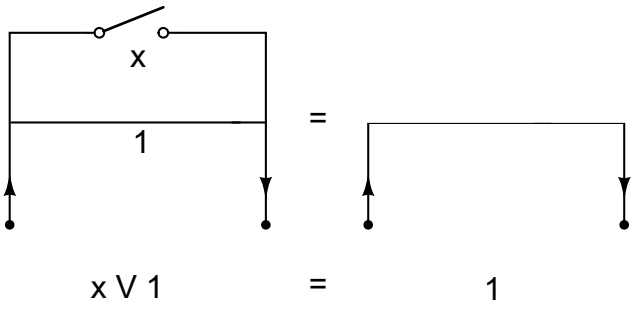
- s postulati:

$$x \vee 1 \overset{P2}{=} (x \vee 1) \cdot 1 \overset{P5}{=} (x \vee 1) \cdot (x \vee \overline{x}) \cdot 1 \overset{P4}{=} x \vee (1 \cdot \overline{x}) \overset{P2}{=} x \vee \overline{x} \overset{P5}{=} 1$$

- s pravilnostno tabelo:

x	$x \vee 1$	1
0	1	1
1	1	1

- s stikalno shemo



Poglavje 3

Preklopne funkcije

Preklopne ali Boolove funkcije tvorimo s pomočjo preklopnih spremenljivk in logičnih operatorjev ($\vee$, $\&$, $^-$, ...). Tako kot preklopne spremenljivke tudi preklopne funkcije lahko zavzamejo samo dve vrednosti - 0 (*ang.* false) ali 1 (*ang.* true).

3.1 Zapis preklopne funkcije s pravilnostno tabelo

Preklopne funkcije lahko zapišemo s pravilnostno tabelo, v kateri so navedene vse kombinacije vhodnih spremenljivk z ustreznimi vrednostmi funkcije. Tabela za preklopno funkcijo z n vhodnimi spremenljivkami ima 2^n vrstic. Obstaja 2^{2^n} različnih preklopnih funkcij z n spremenljivkami.

Primer 5: Pravilnostna tabela za preklopno funkcijo dveh spremenljivk:

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

3.2 Analitični zapis preklopnih funkcij

Preklopne funkcije najpogosteje zapisujemo analitično v

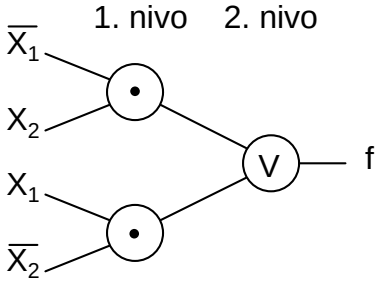
- popolni disjunktivni normalni obliki (PDNO) ali
- popolni konjunktivni normalni obliki (PKNO).

Beseda normalna pomeni, da v funkciji nastopata dva nivoja operatorjev, beseda popolna pa, da vstopajo v prvi nivo operatorjev vse spremenljivke.

Primer 6: Zapis preklopne funkcije iz prejšnjega primera v PDNO in PKNO.

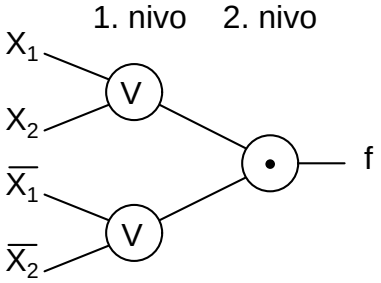
Popolna disjunktivna normalna oblika

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



Popolna konjunktivna normalna oblika

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



3.2.1 Popolna disjunktivna normalna oblika

Osnovni gradnik PDNO je konjunkcija čez vse spremenljivke, ki je operator prvega nivoja. Operator drugega nivoja je disjunkcija.
Konjunkcijo vseh vhodnih spremenljivk imenujemo minterm. Za n vhodnih spremenljivk ga zapišemo kot

$$m_i = x_1^{\omega_{1i}} \cdot x_2^{\omega_{2i}} \cdot \ldots \cdot x_n^{\omega_{ni}} \quad ,$$

kjer je

$$x_k^{\omega_{ki}} = \begin{cases} x_k & \text{za} \quad w_{ki} = 1 \\ \overline{x}_k & \text{za} \quad w_{ki} = 0 \end{cases}$$

in je ω_{ki} vrednost spremenljivke k pri i-ti vhodni kombinaciji. Različnih mintermov je 2^n, toliko kot je kombinacij n vhodnih spremenljivk.

Primer 7: Mintermi, sestavljeni iz dveh vhodnih spremenljivk

i	x_1	x_2	m_i
0	0	0	$m_0 = x_1^0 \cdot x_2^0 = \overline{x}_1 \cdot \overline{x}_2$
1	0	1	$m_1 = x_1^0 \cdot x_2^1 = \overline{x}_1 \cdot x_2$
2	1	0	$m_2 = x_1^1 \cdot x_2^0 = x_1 \cdot \overline{x}_2$
3	1	1	$m_3 = x_1^1 \cdot x_2^1 = x_1 \cdot x_2$

Splošen analitičen zapis preklopne funkcije z n spremenljivkami v PDNO:

$$\begin{aligned} f^{(n)} &= \bigvee_{i=0}^{2^n-1} f_i \cdot m_i \\ &= \bigvee_{i=0}^{2^n-1} f_i \cdot (x_1^{\omega_{1i}} \cdot x_2^{\omega_{2i}} \cdot \ldots \cdot x_n^{\omega_{ni}}) \end{aligned}$$

V zgornjem zapisu je f_i vrednost funkcije pri i-ti kombinaciji vhodnih spremenljivk. Minterm m_i nastopa v zapisu funkcije, če je $f_i = 1$. Povedano drugače, v PDNO preklopne funkcije nastopajo tisti mintermi, pri katerih je vrednost preklopne funkcije 1.

3.2.2 Popolna konjunktivna normalna oblika

Osnovni gradnik PKNO je disjunkcija vseh vhodnih spremenljivk, ki je operator prvega nivoja. Operator drugega nivoja je konjunkcija.
Disjunkcijo vseh vhodnih spremenljivk imenujemo maksterm. Za n vhodnih spremenljivk ga zapišemo kot

$$M_j = x_1^{\omega_{1j}} \vee x_2^{\omega_{2j}} \vee \ldots \vee x_n^{\omega_{nj}} \quad ,$$

pri čemer je

$$j = 2^n - 1 - i$$

obratni indeks ko i. Različnih makstermov je 2^n, toliko kot je kombinacij n vhodnih spremenljivk.

Primer 8: Makstermi, sestavljeni iz dveh vhodnih spremenljivk

i	j	x_1	x_2	M_j
0	3	0	0	$M_3 = x_1^1 \vee x_2^1 = x_1 \vee x_2$
1	2	0	1	$M_2 = x_1^1 \vee x_2^0 = x_1 \vee \overline{x}_2$
2	1	1	0	$M_1 = x_1^0 \vee x_2^1 = \overline{x}_1 \vee x_2$
3	0	1	1	$M_0 = x_1^0 \vee x_2^0 = \overline{x}_1 \vee \overline{x}_2$

Splošen analitičen zapis preklopne funkcije z n spremenljivkami v PKNO:

$$\begin{aligned} f^{(n)} &= \mathop{\&}_{i=0}^{2^n-1} f_i \vee M_j \\ &= \mathop{\&}_{i=0}^{2^n-1} f_i \vee M_{2^n-1-i} \\ &= \mathop{\&}_{i=0}^{2^n-1} f_i \vee (x_1^{\omega_{1j}} \vee x_2^{\omega_{2j}} \vee \ldots \vee x_n^{\omega_{nj}}) \end{aligned}$$

V zgornjem zapisu je f_i vrednost funkcije pri i-ti kombinaciji vhodnih spremenljivk. Maksterm M_j nastopa v zapisu funkcije, če je $f_i = 0$. Povedano drugače, v PKNO preklopne funkcije nastopajo tisti makstermi, pri katerih je vrednost preklopne funkcije 0.

Primer 9: Zapiši preklopno funkcijo podano s spodnjo pravilnostno tabelo v PDNO in PKNO.

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

$\Rightarrow$

i	j	x_1	x_2	m_i	M_j	$f_i = f(x_1,x_2)$
0	3	0	0	m_0	M_3	0
1	2	0	1	m_1	M_2	1
2	1	1	0	m_2	M_1	1
3	0	1	1	m_3	M_0	0

$$\begin{aligned} f_{PDNO} &= m_1 \vee m_2 = \overline{x}_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \\ f_{PKNO} &= M_3 \cdot M_0 = (x_1 \vee x_2) \cdot (\overline{x}_1 \vee \overline{x}_2) \end{aligned}$$

3.2.3 Relacije med mintermi in makstermi

Med mintermi in makstermi veljajo naslednje relacije:

- $\bigvee_{i=0}^{2^n-1} m_i = 1$
- $\mathop{\&}_{j=2^n-1}^{0} M_j = 0$
- $\overline{M}_j = m_i = m_{2^n-1-j}$
- $m_i \vee M_j = m_i \vee M_{2^n-1-i} = 1$
- $m_i \cdot m_j = m_i \cdot m_{2^n-1-i} = 0 \quad , \quad i \neq j$
- $M_i \vee M_j = M_i \vee M_{2^n-1-i} = 1 \quad , \quad i \neq j$

Primer 10: Relacije med mintermi in makstermi za $n = 2$.

x_1	x_2	$\overline{x}_1 \cdot \overline{x}_2$ m_0	$\overline{x}_1 \cdot x_2$ m_1	$x_1 \cdot \overline{x}_2$ m_2	$x_1 \cdot x_2$ m_3	$\bigvee_{i=0}^{3} m_i$	$x_1 \vee x_2$ M_3	$x_1 \vee \overline{x}_2$ M_2	$\overline{x}_1 \vee x_2$ M_1	$\overline{x}_1 \vee \overline{x}_2$ M_0	$\&_{j=3}^{0} M_j$
0	0	**1**	0	0	0	1	**0**	1	1	1	0
0	1	0	**1**	0	0	1	1	**0**	1	1	0
1	0	0	0	**1**	0	1	1	1	**0**	1	0
1	1	0	0	0	**1**	1	1	1	1	**0**	0
						$\|\|$					$\|\|$
						1					0

3.3 Numerični zapis preklopnih funkcij v PDNO in PKNO

- Preklopno funkcijo zapisano analitično v PDNO

  $$f^{(n)} = \bigvee_{i=0}^{2^n-1} f_i \cdot m_i$$

  večkrat podamo v numeričnem zapisu

  $$f^{(n)} = \vee(\ldots, i, \ldots) \quad ,$$

  kjer v oklepaju navedemo samo indekse i tistih mintermov m_i, pri katerih je funkcijska vrednost f_i enaka 1.

- Podobno preklopno funkcijo zapisano analitično v PKNO

  $$f^{(n)} = \mathop{\&}_{i=0}^{2^n-1} f_i \vee M_j$$

  večkrat podamo v numeričnem zapisu

  $$f^{(n)} = \&(\ldots, j, \ldots) \quad ,$$

  kjer v oklepaju navedemo samo indekse j tistih makstermov M_j, pri katerih je funkcijska vrednost f_i enaka 0.

Primer 11: Preklopno funkcijo podano s pravilnostno tabelo

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

lahko zapišemo kot

$$\begin{aligned} f(x_1, x_2) &= f_{PDNO} = m_1 \vee m_2 = \vee(1, 2) \\ &= f_{PKNO} = M_3 \cdot M_0 = \&(3, 0) \quad . \end{aligned}$$

3.4 Pretvarjanje med PDNO in PKNO

Funkcijo podano v PDNO lahko pretvorimo v PKNO in obratno. Postopek:

- Najprej določimo indekse manjkajočih mintermov (oziroma makstermov).
- Nato izvedemo obrate manjkajočih indeksov,

$$j = 2^n - 1 - i \quad .$$

- Preklopno funkcijo zapišemo z makstermi (oziroma mintermi), ki jih določajo obrnjeni indeksi.

Primer 12: Preklopno funkcijo $f^{(3)} = \vee(0, 1, 4, 5, 6)$ pretvorimo v PKNO.
V tabeli mintermov in makstermov prečrtamo minterme, ki nastopajo v preklopni funkciji

m_i	i	~~0~~	~~1~~	2	3	~~4~~	~~5~~	~~6~~	7
M_j	j	7	6	5	4	3	2	1	0

Preklopno funkcijo v PKNO zapišemo s tistimi makstermi, ki nastopajo pri neprečrtanih mintermih,

$$f^{(3)} = \&(5, 4, 0) \quad .$$

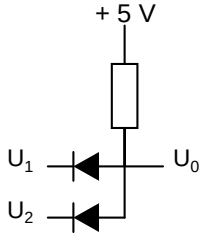
Poglavje 4

Tehnološke rešitve preklopnih funkcij

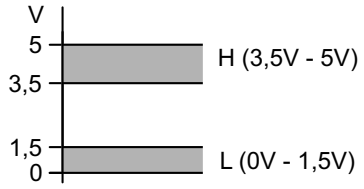
4.1 Polprevodniška vezja

4.1.1 Diodna vezja

Oglejmo si delovanje vezja sestavljenega iz diod in upora, ki je prikazano na spodnji sliki. Kadar je ena od napetosti na diodi (U_1 ali U_2) nizka (L), teče skozi ustrezno diodo tok. Ker tok teče, pride na uporu do padca napetosti, zato je napetost U_0 nizka (L). V primeru, da sta napetosti U_1 in U_2 visoki, tok ne teče in ne pride do padca napetosti, zato je U_0 visoka (H). Delovanje vezja podaja ustrezna tabela napetosti (fizikalna tabela).



U_1	U_2	U_o
L	L	L
L	H	L
H	L	L
H	H	H



Iz fizikalne tabele lahko preidemo v pravilnostno tabelo na dva načina:

- princip pozitivne logike ($L \Rightarrow 0$, $H \Rightarrow 1$)

$x_1 = U_1$	$x_2 = U_2$	$x_1 \cdot x_2 = U_0$
0	0	0
0	1	0
1	0	0
1	1	1

- princip negativne logike ($L \Rightarrow 1$, $H \Rightarrow 0$)

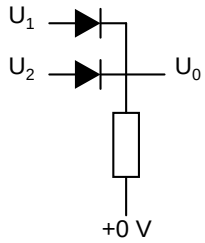
$x_1 = U_1$	$x_2 = U_2$	$x_1 \vee x_2 = U_0$
1	1	1
1	0	1
0	1	1
0	0	0

$\Rightarrow$

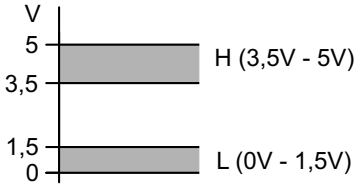
$x_1 = U_1$	$x_2 = U_2$	$x_1 \vee x_2 = U_0$
0	0	0
0	1	1
1	0	1
1	1	1

Pri pozitivni logiki vezje izvaja konjunkcijo, pri negativni logiki pa disjunkcijo.

Na podoben način lahko opišemo tudi delovanje vezja, predstavljenega na spodnji sliki.



U_1	U_2	U_o
L	L	L
L	H	H
H	L	H
H	H	H



Iz fizikalne tabele dobimo naslednji pravilnostni tabeli:

- pozitivna logika (L⇒ 0, H⇒ 1)

$x_1 = U_1$	$x_2 = U_2$	$x_1 \vee x_2 = U_0$
0	0	0
0	1	1
1	0	1
1	1	1

- negativna logika (L⇒ 1, H⇒ 0)

$x_1 = U_1$	$x_2 = U_2$	$x_1 \cdot x_2 = U_0$
1	1	1
1	0	0
0	1	0
0	0	0

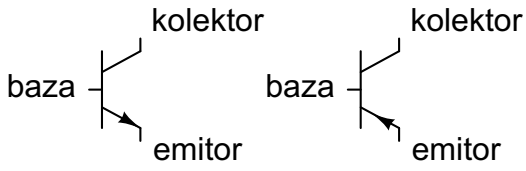
$\Rightarrow$

$x_1 = U_1$	$x_2 = U_2$	$x_1 \cdot x_2 = U_0$
0	0	0
0	1	0
1	0	0
1	1	1

Pri pozitivni logiki vezje izvaja disjunkcijo, pri negativni logiki pa konjunkcijo.

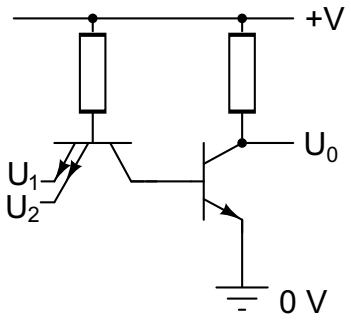
4.1.2 Vezja TTL

Vezja TTL so zasnovana na polprevodniških elementih - NPN in PNP tranzistorjih. TTL je angleška kratica za Transistor Transistor Logic. Pri tranzistorjih napetost med bazo in emitorjem krmili tok med emitorjem in kolektorjem. Kadar je pri tranzistorju NPN napetost med bazo in emitorjem manjša od praga, potem je tranzistor zaprt – tok ne teče, kadar pa je napetost večja od praga pa tok teče.



NPN tranzistor PNP tranzistor

Oglejmo si delovanje vezja, prikazanega na spodnji sliki. Kadar sta obe napetosti U_1 in U_2 visoki, potem je napetost med bazo in emitorjem levega tranzistorja manjša od praga in tranzistor je zaprt. Ker tok ne teče, na tranzistorju ni padca napetosti, zato je na kolektorju visoka napetost. Napetost med bazo in emitorjem desnega tranzistorja je tako večja od praga in desni tranzistor je odprt – tok teče. Ker tok teče, pride na desnem uporu do padca napetosti in na izhodu dobimo nizko napetost. Drugače pa je, kadar je vsaj ena od napetosti U_1 ali U_2 nizka. V tem primeru je odprt levi tranzistor, zato je na njegovem kolektorju nizka napetost. Nizka napetost na bazi desnega tranzistorja slednjega zapre, zato tok ne teče, na desnem uporu ni padca napetosti in izhodna napetost je visoka. Delovanje vezja TTL prikazanega na spodnji sliki podaja ustrezna fizikalna tabela, iz katere lahko hitro razberemo, da v primeru uporabe pozitivne logike vezje izvaja operacijo NAND (negirani in, Sheffer), v primeru negativne logike pa operacijo NOR (negirani ali, Pierce).



U_1	U_2	U_0
L	L	H
L	H	H
H	L	H
H	H	L

Pozitivna logika
$\overline{x_1 \cdot x_2} = x_1 \uparrow x_2$

(NAND ali Sheffer)

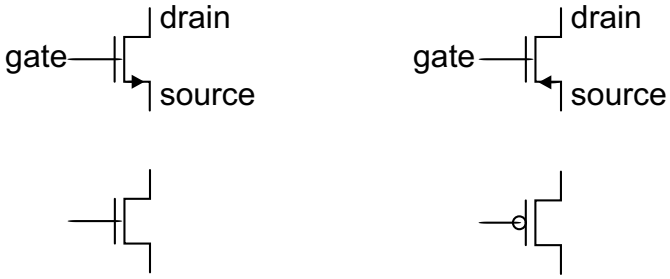
Negativna logika
$\overline{x_1 \vee x_2} = x_1 \downarrow x_2$

(NOR ali Pierce)

4.1.3 Vezja MOS

Najbolj razširjene tehnologija je MOS (*ang.* Metal Oxyde Semiconductor). Poznamo 3 tehnologije: NMOS (*ang.* Negative MOS), ki je zasnovana na tranzistorjih NMOS, PMOS (*ang.* Pozitive MOS), ki je zasnovana na tranzistorjih PMOS in CMOS (*ang.* Complementary MOS), ki vključuje tako NMOS kot PMOS tranzistorje.

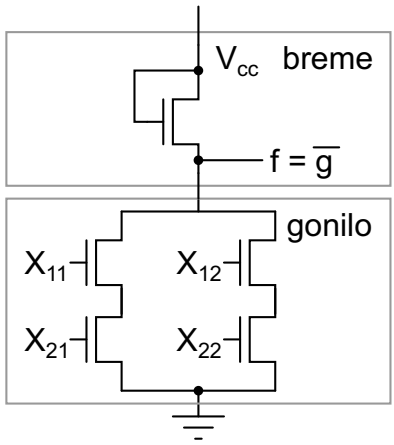
- Tranzistorji NMOS in PMOS:



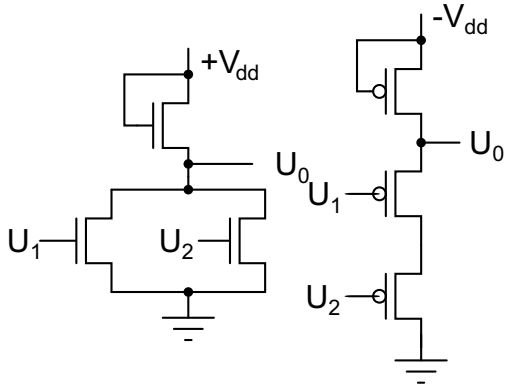
NMOS tranzistor PMOS tranzistor

Za NMOS tranzistorje velja, da tok iz priključka drain (*ang.* ponor) teče proti priključku source (*ang.* vir) kadar je napetost med priključkoma gate (*ang.* vrata) in source višja od praga. Za PMOS tranzistorje velja, da tok iz priključka source teče proti priključku drain, kadar je napetost med priključkoma gate in source nižja od praga.

- MOS vezja sestavlja breme (*ang.* load) in gonilo (*ang.* driver). Izhodna funkcija f je vedno negacija funkcije gonila g.



- Paralelna (vzporedna) vezava NMOS tranzistorjev in serijska (zaporedna) vezava PMOS tranzistorjev:



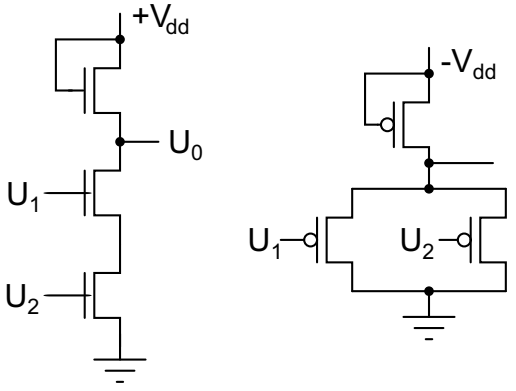
Fizikalna tabela

U_1	U_2	U_0
L	L	H
L	H	L
H	L	L
H	H	L

Pozitivna logika: $\overline{x_1 \vee x_2} = x_1 \downarrow x_2$ (NOR ali Pierce)

Negativna logika: $\overline{x_1 \cdot x_2} = x_1 \uparrow x_2$ (NAND ali Sheffer)

- Serijska (zaporedna) vezava NMOS tranzistorjev in paralelna (vzporedna) vezava PMOS tranzistorjev:



Fizikalna tabela

U_1	U_2	U_0
L	L	H
L	H	H
H	L	H
H	H	L

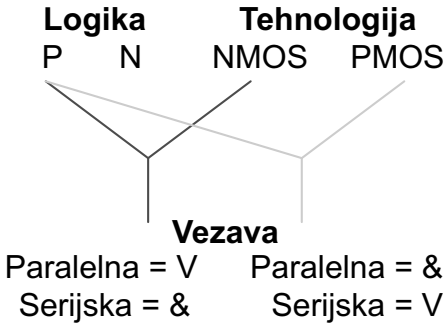
Pozitivna logika: $\overline{x_1 \cdot x_2} = x_1 \uparrow x_2$ (NAND ali Sheffer)

Negativna logika: $\overline{x_1 \vee x_2} = x_1 \downarrow x_2$ (NOR ali Pierce)

- Na funkcijo vplivajo naslednji parametri:
  - logika (pozitivna - P, negativna - N)
  - vezava (serijska - S, paralelna - P)
  - tehnologija (PMOS - P, NMOS - N)

logika	vezava	tehnologija	funkcija
P	S	P	$\downarrow$
P	S	N	$\uparrow$
P	P	P	$\uparrow$
P	P	N	$\downarrow$
N	S	P	$\uparrow$
N	S	N	$\downarrow$
N	P	P	$\downarrow$
N	P	N	$\uparrow$

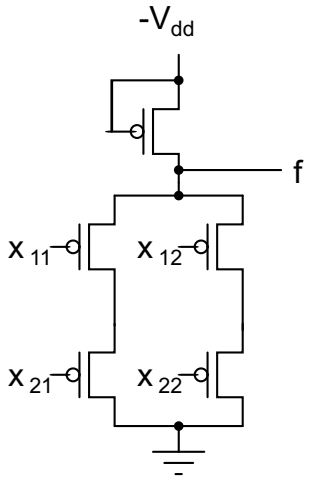
Vsaka sprememba parametra spremeni funkcijo, dve spremembi pa jo ohranjata.



.

- Kompleksne MOS celice dobimo, če v gonilu dopuščamo serijsko-paralelne ali paralelno-serijske vezave tranzistorjev.

  – Paralelno-serijska vezava PMOS tranzistorjev



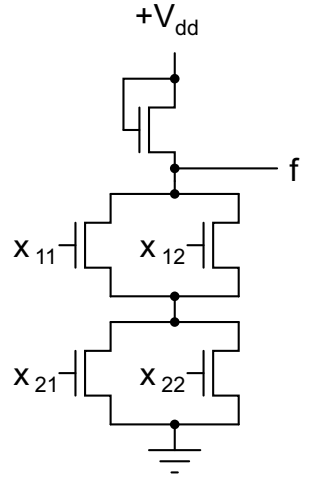
Pozitivna logika:

$$f = \overline{\underset{j}{\&}(\underset{i}{\vee} x_{ij})}$$

Negativna logika:

$$f = \overline{\underset{j}{\vee}(\underset{i}{\&} x_{ij})}$$

  – Serijsko-paralelna vezava NMOS tranzistorjev



Pozitivna logika:

$$f = \overline{\underset{i}{\&}(\underset{j}{\vee} x_{ij})}$$

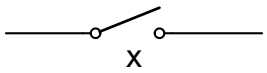
Negativna logika:

$$f = \overline{\underset{i}{\vee}(\underset{j}{\&} x_{ij})}$$

4.2 Stikalna (kontaktna) vezja

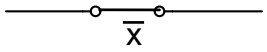
S stikalnimi vezji lahko realiziramo poljubno preklopno funkcijo. Glede na položaje stikal lahko tok steče iz enega konca vezja na drugi konec ali pa ne. V prvem primeru vzamemo, da ima preklopna funkcija vrednost 1, v drugem pa, da ima vrednost 0.

- spremenljivka x

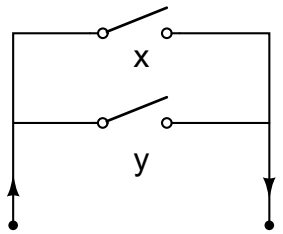
  Ko je $x = 0$ je stikalo odprto (razklenjeno), ko je $x = 1$ pa je stikalo zaprto (sklenjeno).

- spremenljivka $\overline{x}$

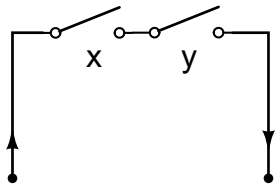
  

  Ko je $x = 0$ je stikalo zaprto (sklenjeno), ko je $x = 1$ pa je stikalo odprto (razklenjeno).

- disjunkcijo rišemo kot paralelno vezavo stikal, konjunkcijo pa kot serijsko vezavo stikal

  $x \vee y$

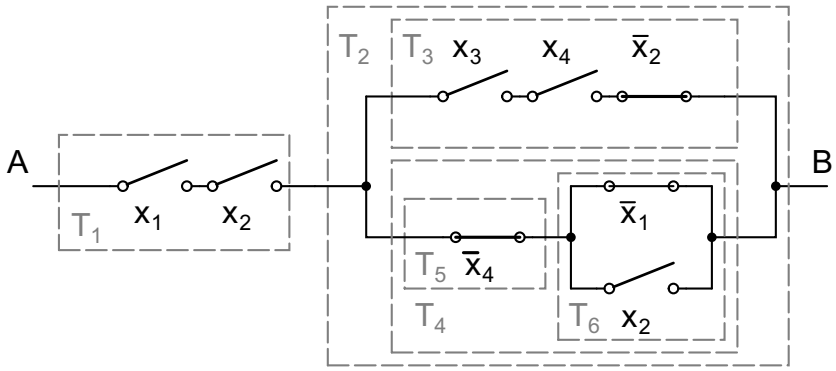
  $x \cdot y$

Analiza stikalnih vezij

Namen analize stikalnega vezja je, da iz obstoječega vezave stikal razberemo preklopno funkcijo, ki jo vezje predstavlja.

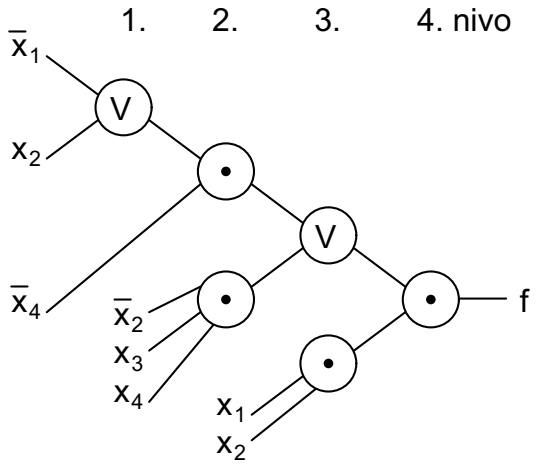
- V enostavnih primerih lahko ugotovimo funkcijo kontaktnega vezja z združevanjem serijskih in paralelnih vezav.

  Primer 13: Zapišimo funkcijo stikalnega vezja na sliki:



$$\begin{aligned} f_{AB} &= T_1 \cdot T_2 = T_1 \cdot (T_3 \vee T_4) = T_1 \cdot (T_3 \vee T_5 \cdot T_6) \\ &= (x_1 \cdot x_2) \cdot (\overline{x}_2 x_3 x_4 \vee \overline{x}_4 \cdot (\overline{x}_1 \vee x_2)) \end{aligned}$$

  Grafični prikaz funkcije:

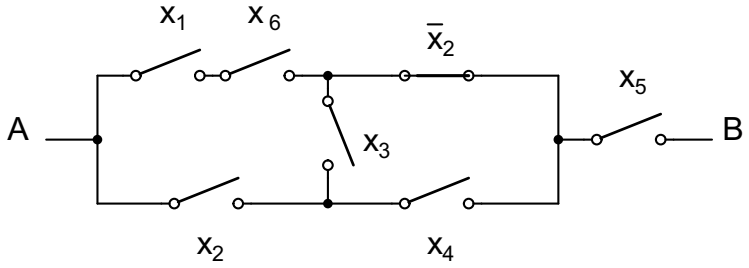


- Včasih pa imamo vezje, kjer povezave stikal niso čisto serijske oziroma paralelne. Tako so na primer mostična vezja. V takem primeru določimo funkcijo s pomočjo:

  – Povezovalnih množic (*ang.* tie-sets):
  Povezovalna množica določa spremenljivke, ki sklenejo točki A in B tako, da že odprtje ene same spremenljivke odpre tudi pot A-B. Poiskati moramo vse možne povezovalne množice. Preklopno funkcijo vezja dobimo tako, da spremenljivke posameznih množic povežemo konjunktivno, dobljene konjunktivne izraze pa med seboj disjunktivno.

– Prekinjevalnih množic (*ang.* cut-sets)
Prekinjevalna množica določa spremenljivke, ki odprejo pot A-B, če so oprete vse spremenljivke v množici. Zaprtje ene same pa zopet poveže točki A in B skupaj. Poiskati moramo vse možne prekinjevalne množice. Preklopno funkcijo vezja dobimo tako, da spremenljivke posameznih množic povežemo disjunktivno, dobljene disjunktivne izraze pa med seboj konjunktivno.

Primer 14: Zapišimo funkcijo stikalnega vezja na sliki



- Povezovalne množice
  - $x_1, x_6, \overline{x_2}, x_5$
  - x_2, x_4, x_5
  - x_1, x_6, x_3, x_4, x_5
  - $x_2, x_3, \overline{x}_2, x_5$

$$\begin{aligned} f_{AB} &= x_1\overline{x}_2x_5x_6 \vee x_1x_3x_4x_5x_6 \vee \underbrace{x_2\overline{x}_2x_3x_5}_{0} \vee x_2x_4x_5 \\ &= x_1\overline{x}_2x_5x_6 \vee x_1x_3x_4x_5x_6 \vee x_2x_4x_5 \end{aligned}$$

- Prekinjevalne množice:
  - x_1, x_2
  - x_6, x_2
  - x_1, x_3, x_4
  - x_6, x_3, x_4
  - $\overline{x}_2, x_3, x_2$
  - $\overline{x}_2, x_4$
  - x_5

$$\begin{aligned} f_{AB} &= (x_1 \vee x_2) \cdot (x_2 \vee x_6) \cdot (x_1 \vee x_3 \vee x_4) \cdot (x_3 \vee x_4 \vee x_6) \cdot \underbrace{(x_2 \vee \overline{x}_2 \vee x_3)}_{1} \cdot (\overline{x}_2 \vee x_4) \cdot x_5 \\ &= (x_1 \vee x_2) \cdot (x_2 \vee x_6) \cdot (x_1 \vee x_3 \vee x_4) \cdot (x_3 \vee x_4 \vee x_6) \cdot (\overline{x}_2 \vee x_4) \cdot x_5 \end{aligned}$$

Poglavje 5

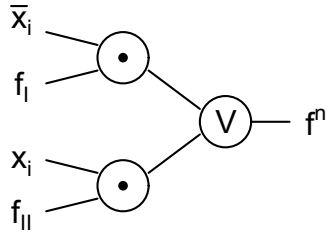
Razčlenjevanje preklopnih funkcij

To je postopek, s katerim želimo iz preklopne funkcije izpostaviti določene spremenljivke tako, da prikažemo vse možne kombinacije izpostavljenih spremenljivk skupaj z ostanki funkcije oziroma ostalih spremenljivk.
Če razčlenjujemo po vseh spremenljivkah je rezultat PDNO ali PKNO.

5.1 Shanonov teorem

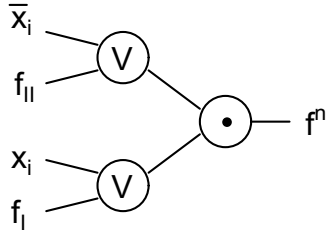
Shanonov teorem določa razčlenitev preklopne funkcije po eni spremenljivki

$$\begin{aligned} f^{(n)} &= \overline{x}_i \cdot f_I \vee x_i \cdot f_{II} \\ & f_I = f^{(n)}(x_1, x_2, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \\ & f_{II} = f^{(n)}(x_1, x_2, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) \end{aligned}$$



Dualni zapis:

$$\begin{aligned} f^{(n)} &= (\overline{x}_i \vee f_{II}) \cdot (x_i \vee f_I) \\ & f_I = f^{(n)}(x_1, x_2, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \\ & f_{II} = f^{(n)}(x_1, x_2, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) \end{aligned}$$



5.2 Razčlenjevanje po k spremenljivkah

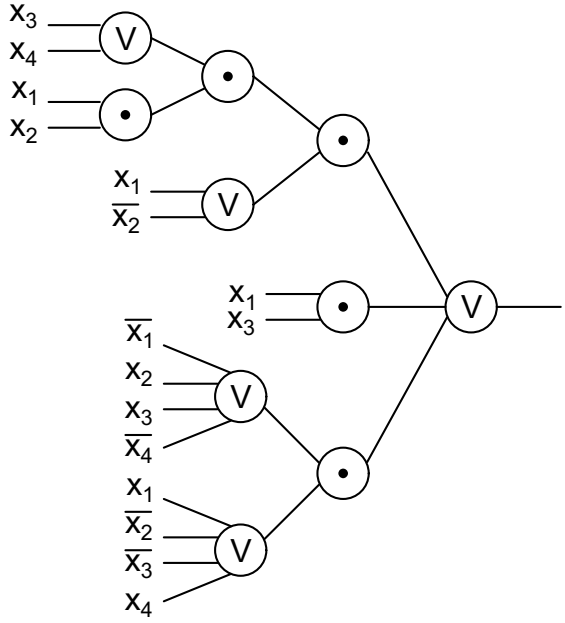
Splošna oblika razčlenjevanja po k spremenljivkah. Za spremenljivke ni nujno, da so zaporedne!

$$f^{(n)} = \bigvee_{i=0}^{2^k-1} x_1^{w_{i1}} \cdot x_2^{w_{i2}} \cdot \ldots \cdot x_k^{w_{ik}} \cdot f(\overbrace{w_{i1}, w_{i2}, \ldots, w_{ik}}^{k}, x_{k+1}, \ldots, x_n)$$

$$f^{(n)} = \bigwedge_{i=2^k-1}^{0} (x_1^{w_{i1}} \vee x_2^{w_{i2}} \vee \ldots \vee x_k^{w_{ik}} \vee f(\overbrace{\overline{w}_{i1}, \overline{w}_{i2}, \ldots, \overline{w}_{ik}}^{k}, x_{k+1}, \ldots, x_n))$$

Primer 15: Razčlenimo dano funkcijo po spremenljivkah x_1 in x_3 na oba načina.

$$f^{(4)} = [x_1x_2(x_3 \vee x_4)]\,(x_1 \vee \overline{x}_2) \vee x_1x_3 \vee [(\overline{x}_1 \vee x_2 \vee x_3 \vee \overline{x}_4) \cdot (x_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee x_4)]$$



- Razčlenitev po x_1 in x_3 v smeri PDNO

$$\begin{aligned} f^{(4)}_{PDNO} &= \overline{x}_1\overline{x}_3 \cdot f(0, x_2, 0, x_4) \vee \\ &\quad \overline{x}_1x_3 \cdot f(0, x_2, 1, x_4) \vee \\ &\quad x_1\overline{x}_3 \cdot f(1, x_2, 0, x_4) \vee \\ &\quad x_1x_3 \cdot f(1, x_2, 1, x_4) \\ &= \overline{x}_1\overline{x}_3 \cdot 1 \vee \overline{x}_1x_3(\overline{x}_2 \vee x_4) \vee x_1\overline{x}_3(x_2 \vee \overline{x}_4) \vee x_1x_3 \cdot 1 \end{aligned}$$

- Razčlenitev po x_1 in x_3 v smeri PKNO

$$\begin{aligned} f^{(4)}_{PKNO} &= (\overline{x}_1 \vee \overline{x}_3 \vee f(1, x_2, 1, x_4)) \cdot \\ &\quad (\overline{x}_1 \vee x_3 \vee f(1, x_2, 0, x_4)) \cdot \\ &\quad (x_1 \vee \overline{x}_3 \vee f(0, x_2, 1, x_4)) \cdot \\ &\quad (x_1 \vee x_3 \vee f(0, x_2, 0, x_4)) \\ &= (\underbrace{\overline{x}_1 \vee \overline{x}_3 \vee 1}_{1}) \cdot (\overline{x}_1 \vee x_3 \vee x_2 \vee \overline{x}_4) \cdot (x_1 \vee \overline{x}_3 \vee \overline{x}_2 \vee x_4) \cdot (\underbrace{x_1 \vee x_3 \vee 1}_{1}) \end{aligned}$$

Poglavje 6

Elementarni operatorji Boolove logike

Za razliko od klasične algebre pri preklopni logiki velja, da je število različnih funkcij odvisno od števila vhodnih spremenljivk. Veljajo naslednje relacije:

	$\overbrace{\quad\quad\quad}^{n}$				$\overbrace{\quad\quad\quad\quad\quad\quad\quad}^{2^{2^n}}$			
	x_1	x_2	$\dots$	x_n	f_0	f_1	$\dots$	$f_{2^{2^n}-1}$
2^n	0	0	$\dots$	0				
	0	0	$\dots$	1				
			$\vdots$					
	1	1	$\dots$	0				
	1	1	$\dots$	1				

Kot je razvidno iz tabele imamo za n spremenljivk 2^n kombinacij in 2^{2^n} različnih funkcij.

6.1 Operatorji nad dvema spremenljivkama

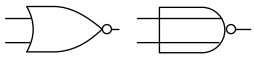
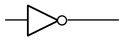
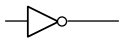
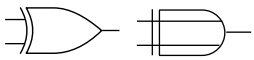
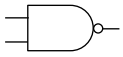
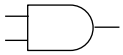
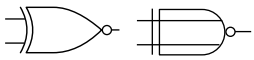
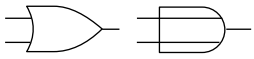
Čeprav je število različnih funkcij lahko zelo veliko, pa v praksi uporabljamo za operatorje nad spremenljivkami le tiste, ki izhajajo iz tabele funkcij nad dvema spremenljivkama.

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
		0	$\downarrow$		$\overline{x}_1$		$\overline{x}_2$	$\bigtriangledown$	$\uparrow$	&	$\equiv$	x_2	$\rightarrow$	x_1	$\leftarrow$	V	1
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Uporabljamo torej le $2^{2^2} = 16$ operatorjev, ki jih delimo v tri skupine.

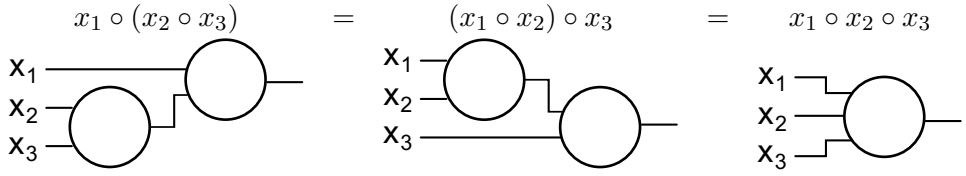
- trivialni operatorji (0, 1, x_1, $\overline{x}_1$, x_2, $\overline{x}_2$)
- logična vrata (&, V, $\uparrow$, $\downarrow$)
- XOR-izpeljane funkcije ($\bigtriangledown$, $\equiv$, $\rightarrow$ in ostale)

Zapis preklopnih funkcij s standardnim naborom operatorjev in njihovi logični simboli:

funkcija	oznaka	pomen	opis simbol
f_0	0	0	konstanta 0
f_1	$\downarrow$	$x_1 \downarrow x_2 = \overline{x_1 \vee x_2}$	ne ali, NOR, Pierce 
f_2		$\overline{x_2 \rightarrow x_1} = \overline{\overline{x}_2 \vee x_1} = \overline{x}_1 \cdot x_2$	negacija implikacije
f_3	$\bar{}$	$\overline{x}_1$	negacija, NOT 
f_4		$\overline{x_1 \rightarrow x_2} = \overline{\overline{x}_1 \vee x_2} = x_1 \cdot \overline{x}_2$	negacija implikacije
f_5	$\bar{}$	$\overline{x}_2$	negacija, NOT 
f_6	$\triangledown$	$x_1 \triangledown x_2 = \overline{x}_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2$ vsota po modulu 2	izključni (ekskluzivni) ali, XOR (EXOR), 
f_7	$\uparrow$, $\mid$	$x_1 \uparrow x_2 = \overline{x_1 \cdot x_2}$	ne in, NAND, Sheffer 
f_8	&, $\cdot$	$x_1 \cdot x_2 = x_1 x_2$	in, AND 
f_9	$\equiv$	$x_1 \equiv x_2 = \overline{x}_1 \overline{x}_2 \vee x_1 x_2 = \overline{x_1 \triangledown x_2}$	ekvivalenca, XNOR 
f_{10}		x_2	spremenljivka
f_{11}	$\rightarrow$	$x_1 \rightarrow x_2 = \overline{x}_1 \vee x_2$	implikacija
f_{12}		x_1	spremenljivka
f_{13}	$\rightarrow$	$x_1 \leftarrow x_2 = x_2 \rightarrow x_1 = x_1 \vee \overline{x}_2$	implikacija
f_{14}	$\vee$	$x_1 \vee x_2$	ali, OR 
f_{15}	1	1	konstanta 1

6.1.1 Teorem asociativnosti

Za nekatere operatorje nad dvema spremenljivkama velja asociativnost. Če splošen operator označimo z $\circ$, potem velja:



6.2 Ugotavljanje funkcijske polnosti

Elementarni sistem operatorjev tvorijo operatorji disjunkcije ($\vee$), konjunkcije ($\&$, $\cdot$) in negacije ($^-$), saj z njimi lahko izrazimo poljubno preklopno funkcijo.

Zanima nas, ali lahko tudi z drugimi operatorji opišemo poljubno preklopno funkcijo. Sistem operatorjev je funkcijsko poln, če to lahko storimo, sicer ni.

Funkcijsko polnost ugotavljamo na dva načina:

- s prehodom na elementarni sistem operatorjev ($\vee$, $\&$, $^-$)
- s pomočjo zaprtih razredov T_0, T_1, S, L in M.

6.2.1 Prehod na elementarni sistem operatorjev

Pri tem postopku poskušamo samo z uporabo opazovanih operatorjev zapisati elementarne operatorje. Če nam to uspe, je nabor opazovanih operatorjev funkcijsko poln, drugače pa ne.

Primer 16: Za vse spodaj navedene sisteme operatorjev je dokazano, da predstavljajo funkcijsko polne sisteme.

- Sistem operatorjev ($\vee$, $^-$)
  - negacija: vključena v sistem operatorjev
  - disjunkcija: vključena v sistem operatorjev
  - konjunkcija: $x_1 \cdot x_2 = \overline{\overline{x}_1 \vee \overline{x}_2}$
- Sistem operatorjev ($\&$, $^-$)
  - negacija: vključena v sistem operatorjev
  - konjunkcija: vključena v sistem operatorjev
  - disjunkcija: $x_1 \vee x_2 = \overline{\overline{x}_1 \cdot \overline{x}_2}$
- Operator ($\uparrow$)
  - negacija: $x \uparrow x = \overline{x \cdot x} = \overline{x}$
  - disjunkcija: $(x_1 \uparrow x_1) \uparrow (x_2 \uparrow x_2) = \overline{x}_1 \uparrow \overline{x}_2 = \overline{\overline{x}_1 \cdot \overline{x}_2} = x_1 \vee x_2$
  - konjunkcija: $(x_1 \uparrow x_2) \uparrow (x_1 \uparrow x_2) = \overline{x_1 \uparrow x_2} = \overline{\overline{x_1 \cdot x_2}} = x_1 \cdot x_2$

- Operator ($\downarrow$)
  - negacija: $x \downarrow x = \overline{x \vee x} = \overline{x}$
  - disjunkcija: $(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2) = \overline{x_1 \downarrow x_2} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2$
  - konjunkcija: $(x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2) = \overline{x}_1 \downarrow \overline{x}_2 = \overline{\overline{x}_1 \vee \overline{x}_2} = x_1 \cdot x_2$
- Sistem operatorjev ($\equiv$, $\vee$, 0)
  - negacija: $0 \equiv x = \overline{x}$
  - disjunkcija: vključena v sistem operatorjev
  - konjunkcija: $0 \equiv ((0 \equiv x_1) \vee (0 \equiv x_2)) = \overline{\overline{x}_1 \vee \overline{x}_2} = x_1 \cdot x_2$
- Sistem operatorjev ($\triangledown$, &, 1)
  - negacija: $1 \triangledown x = \overline{x}$
  - konjunkcija: vključena v sistem operatorjev
  - disjunkcija: $1 \triangledown ((1 \triangledown x_1) \cdot (1 \triangledown x_2)) = \overline{\overline{x}_1 \cdot \overline{x}_2} = x_1 \vee x_2$

6.2.2 Zaprti razredi

Pri ugotavljanju funkcijske polnosti s pomočjo zaprtih razredov oziroma lastnosti funkcij, ki omejujejo sistem, bomo uporabili grafično predstavitev logičnih funkcij – Veitchove diagrame.

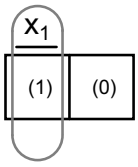
Veitchovi diagrami

Veitchov diagram je raster z 2^n kvadrati, pri čemer je n število spremenljivk. Vsak kvadrat predstavlja en minterm – eno kombinacijo spremenljivk v pravilnostni tabeli. V Veitchovem diagramu so spremenljivke označene tako, da

- vsaka spremenljivka pokriva polovico diagrama,
- vsak presek dveh spremenljivk pa pokriva četrtino diagrama.

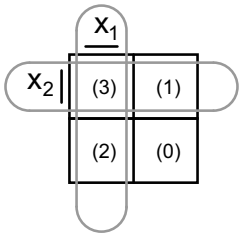
Pri označevanju Veitchovega diagrama velja dogovor, da spremenljivke dodajamo v nasprotni smeri urinega kazalca: začnemo označevati na vrhu levo, nato levo zgoraj, nato spodaj, zamaknjeno za en stolpec z leve, nato desno, zamaknjeno za eno vrstico od vrha, ...

- Za $n = 1$ je število kvadratov $= 2^1 = 2$



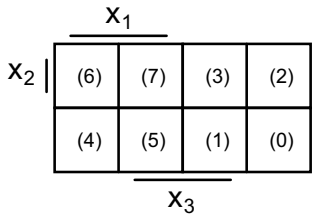
x_1	indeks i minterma m_i
0	0
1	1

- Za $n = 2$ je število kvadratov $= 2^2 = 4$



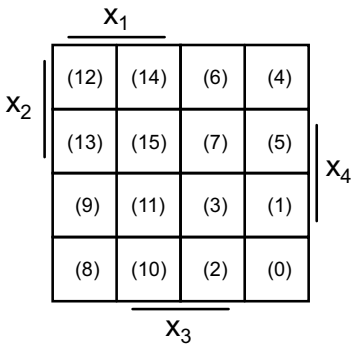
x_1	x_2	indeks i minterma m_i
0	0	0
0	1	1
1	0	2
1	1	3

- Za $n = 3$ je število kvadratov $= 2^3 = 8$



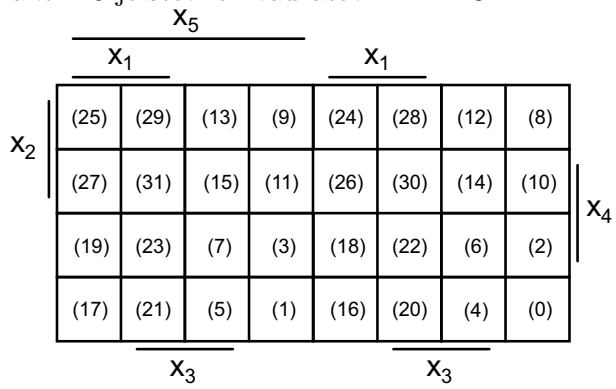
x_1	x_2	x_3	indeks i minterma m_i
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- Za $n = 4$ je število kvadratov $= 2^4 = 16$



x_1	x_2	x_3	x_4	indeks i minterma m_i
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	0
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

- Za $n = 5$ je število kvadratov $= 2^5 = 32$



Določanje funkcijske polnosti sistema operatorjev z zaprtimi razredi

Za vsak operator iz podanega nabora, ugotovimo katere pogoje, ki so predstavljeni z zaprtimi razredi, izpolnjuje. Vsak pogoj predstavlja slabo lastnost, oziroma neke vrste past. Če so vsi operatorji iz nabora ujeti v vsaj eno izmed pasti sistem ni funkcijsko poln. Zaprti razredi so naslednji

- T_0 – razred konstante 0: $f \in T_0$, če $f(0,0,\ldots,0) = 0$
- T_1 – razred konstante 1: $f \in T_1$, če $f(1,1,\ldots,1) = 1$
- S – razred sebidualnih funkcij: $f \in S$, če $f(x_1,x_2,\ldots,x_n) = f^d = \overline{f}(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n)$
- L – razred linearnih funkcij: $f \in L$, če $f(x_1,x_2,\ldots,x_n) = a_0 \bigtriangledown a_1 \cdot x_1 \bigtriangledown \ldots \bigtriangledown a_n \cdot x_n$, pri čemer je $a_i \in 0,1$.
- M – razred monotonih funkcij: $f \in M$, če pri $\omega_i < \omega_j$ velja $f(\omega_i) \leq f(\omega_j)$, pri čemer sta ω_i in ω_j različna in se razlikujeta samo po eni spremenljivki.

Sebidualnost

Primer 17: Dualnost konjunkcije in disjunkcije ter Shefferjevega in Pierceovega operatorja

$$\begin{aligned} f_1(x_1,x_2) &= x_1 \cdot x_2 \\ f_1^d(x_1,x_2) &= \overline{\overline{x}_1 \cdot \overline{x}_2} = \overline{\overline{x}}_1 \vee \overline{\overline{x}}_2 = x_1 \vee x_2 \\ f_2(x_1,x_2) &= x_1 \uparrow x_2 \\ f_2^d(x_1,x_2) &= \overline{\overline{x}_1 \uparrow \overline{x}_2} = \overline{x}_1 \cdot \overline{x}_2 = \overline{x_1 \vee x_2} = x_1 \downarrow x_2 \end{aligned}$$

Sebidualnost enostavno ugotavljamo grafično s pomočjo pravilnostne tabele: primerjamo prvo in zadnjo vrstico, drugo in predzadnjo vrstico, ... Če so vrednosti vseh primerjanih vrstic različne, je funkcija sebidualna.

Primer 18: Vse sebidualne funkcije dveh spremenljivk
Dobimo jih tako, da v zgornji del tabele napišemo vse možne kombinacije, nato pa v spodnji del preslikamo negirane vrednosti.

x_1	x_2	f_1	f_2	f_3	f_4
0	0	0	0	1	1
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	0	0

$$\begin{aligned} f_1 &= x_1, & f_1^d &= \overline{\overline{x}}_1 = x_1 \\ f_2 &= x_2, & f_2^d &= \overline{\overline{x}}_2 = x_2 \\ f_3 &= \overline{x}_2, & f_3^d &= \overline{\overline{x}}_2 = \overline{x}_2 \\ f_4 &= \overline{x}_1, & f_4^d &= \overline{\overline{x}}_1 = \overline{x}_1 \end{aligned}$$

Linearnost

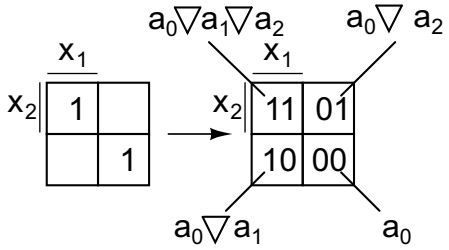
Linearnost preklopnih funkcij lahko določamo z uporabo pravilnostne tabele ali pa grafično s pomočjo Veitchovih diagramov.

Primer 19: Linearnost ekvivalence $f(x_1,x_2) = x_1 \equiv x_2 = \overline{x}_1\overline{x}_2 \vee x_1x_2$

Za vsako kombinacijo spremenljivk lahko zapišemo ustrezno linearno enačbo. Če iz dobljenih linearnih enačb uspemo določi konstante a_i tako, da vse enačbe veljajo, je funkcija linearna:

x_1	x_2	$x_1 \bigtriangledown x_2$	$x_1 \equiv x_2$
0	0	1	$a_0 \bigtriangledown a_1 \cdot 0 \bigtriangledown a_2 \cdot 0 = a_0 = 1$
0	1	0	$a_0 \bigtriangledown a_1 \cdot 0 \bigtriangledown a_2 \cdot 1 = a_0 \bigtriangledown a_2 = 0$
1	0	0	$a_0 \bigtriangledown a_1 \cdot 1 \bigtriangledown a_2 \cdot 0 = a_0 \bigtriangledown a_1 = 0$
1	1	1	$a_0 \bigtriangledown a_1 \cdot 1 \bigtriangledown a_2 \cdot 1 = a_0 \bigtriangledown a_1 \bigtriangledown a_2 = 1$

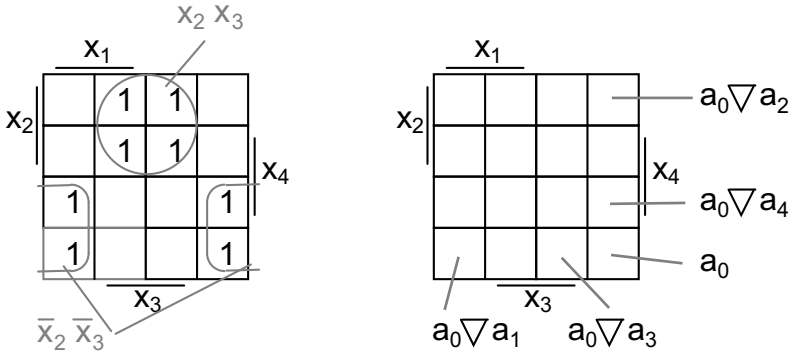
Enačbe za konstante a_i lahko enostavno nastavimo tudi z uporabo Veitchovega diagrama:



$a_0 = 1$
$a_0 \bigtriangledown a_2 = 0$
$a_0 \bigtriangledown a_1 = 0$
$a_0 \bigtriangledown a_1 \bigtriangledown a_2 = 1$

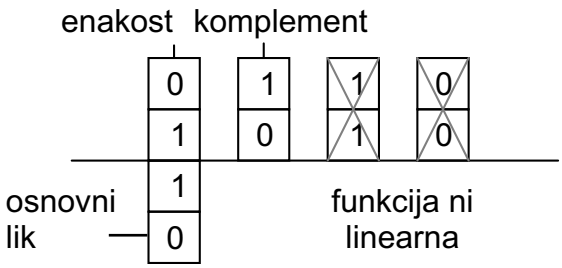
Z upoštevanjem relacij, ki veljajo za operator XOR: $0 \bigtriangledown a = a$, $1 \bigtriangledown a = \overline{a}$, $a \bigtriangledown a = a \cdot \overline{a} \vee \overline{a} \cdot a = 0$ in $a \bigtriangledown a \bigtriangledown a = a \bigtriangledown (a \bigtriangledown a) = a \bigtriangledown 0 = a$ pridemo do naslednjih vrednosti konstant: $a_0 = 1$, $a_1 = 1$ in $a_2 = 1$. Ker veljajo vse štiri enačbe, je funkcija linearna in jo zato lahko zapišemo tudi kot $f(x_1, x_2) = x_1 \equiv x_2 = a_0 \bigtriangledown a_1 x_1 \bigtriangledown a_2 x_2 = 1 \bigtriangledown x_1 \bigtriangledown x_2$.

Primer 20: Linearnost funkcije $f(x_1, x_2, x_3, x_4) = x_2 x_3 \vee \overline{x}_2 \overline{x}_3$
Funkcijo vpišemo v Veitchov diagram

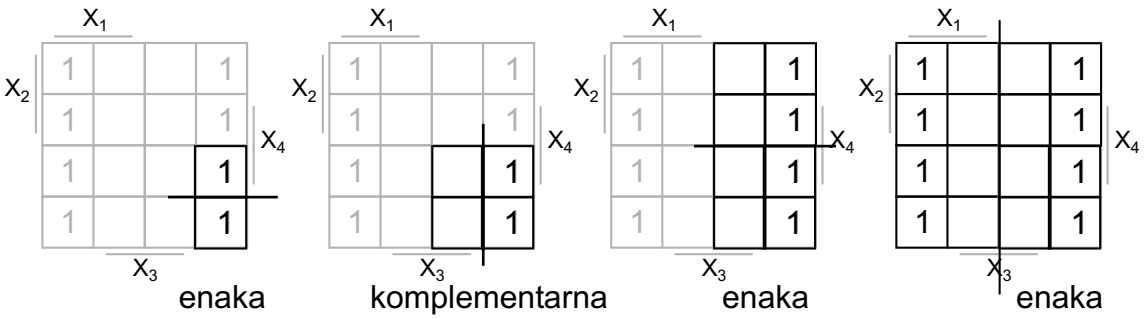


Za določitev konstant a_i potrebujemo 5 enačb. Najbolje je, če vzamemo enačbe, v katerih nastopata največ dve spremenljivki. Ustrezni kvadrati v Veitchovem diagramu so prikazani na zgornji sliki. Iz enačb $a_0 = 1$, $a_0 \bigtriangledown a_1 = 1$, $a_0 \bigtriangledown a_2 = 0$, $a_0 \bigtriangledown a_3 = 0$ in $a_0 \bigtriangledown a_4 = 1$ enostavno določimo vrednosti konstant: $a_0 = 1$, $a_1 = 0$, $a_2 = 1$, $a_3 = 1$ in $a_4 = 0$. Ker so konstante določene tako, da je pravilnih tudi vseh ostalih 11 enačb, na primer, $a_0 \bigtriangledown a_1 \bigtriangledown a_2 \bigtriangledown a_3 \bigtriangledown a_4 = 1 \bigtriangledown 0 \bigtriangledown 1 \bigtriangledown 1 \bigtriangledown 0 = 1$, je funkcija linearna.
Linearnost funkcije lahko preverimo tudi z grafičnim postopkom podvajanja likov:

- Začnemo s kvadratom, ki označuje minterm m_0.
- Kvadrat nato postopno podvajamo: najprej s kvadratom v smeri spremenljivke x_n, nato s pravokotnikom v smeri spremenljivke x_{n-1}, vse do spremenljivke x_1.
- Če so v vseh dodanih likih vrednosti zrcalno enake ali zrcalno komplementarne osnovnemu liku, je funkcija linearna. Primerjanje likov je ilustrirano na spodnji sliki.



Primer 21: Grafično določanje linearnosti za funkcijo $f(x_1, x_2, x_3, x_4) = \overline{x}_3$



Ker so v dodanih likih vrednosti funkcije zracalno enake oziroma komplementarne, je funkcija $f(x_1, x_2, x_3, x_4) = \overline{x}_3$ linearna.

Monotonost

Preklopno funkcijo zapišemo v pravilnostno tabelo, nato pa za vsako kombinacijo $w_i < w_j$, $i \neq j$, preverimo, če velja $f(w_i) \leq f(w_j)$. Preverimo monotonost preklopne funkcije $f^3 = \vee(3, 6, 7)$.

i	ω_i	x_1	x_2	x_3	f^3		
0	ω_0	0	0	0	0	$\omega_0 \leq^? \omega_1, \omega_2, \omega_4$	$0 \leq 0, 0, 0$
1	ω_1	0	0	1	0	$\omega_1 \leq^? \omega_3, \omega_5$	$0 \leq 1, 0$
2	ω_2	0	1	0	0	$\omega_2 \leq^? \omega_3, \omega_6$	$0 \leq 1, 1$
3	ω_3	0	1	1	1	$\omega_3 \leq^? \omega_7$	$1 \leq 1$
4	ω_4	1	0	0	0	$\omega_4 \leq^? \omega_5, \omega_6$	$0 \leq 0, 1$
5	ω_5	1	0	1	0	$\omega_5 \leq^? \omega_7$	$0 \leq 1$
6	ω_6	1	1	0	1	$\omega_6 \leq^? \omega_7$	$1 \leq 1$
7	ω_7	1	1	1	1		

Ker veljajo vse relacije, je funkcija $f^3 = \vee(3, 6, 7)$ monotona.

Primer 22: Z uporabo zaprtih razredov ugotovimo, ali je sistem operatorjev $(x_1 \rightarrow x_2, 0)$ funkcijsko poln.
Pravilnostni tabeli obeh operatorjev in pripadnost zaprtim razredom:

x_1	x_2	$x_1 \rightarrow x_2$	0
0	0	1	0
0	1	1	0
1	0	0	0
1	1	1	0

	T_0	T_1	S	L	M
$\rightarrow$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$
0	$\in$	$\notin$	$\notin$	$\in$	$\in$

Ker za vsak zaprti razred obstaja operator, ki mu ne pripada, je sistem funkcijsko poln.

6.3 Realizacija preklopnih funkcij z vezji nizke stopnje integracije

Za realizacijo preklopnih funkcij so posebno pomembna logična vrata, ki jih zato lahko kupimo kot vezja (čipe) nizke stopnje integracije – SSI (*ang.* Small Scale Integration). Z logičnimi vrati ($\&, \bigvee, \uparrow, \downarrow$) je mogoče na 8 normalno (dvonivojsko) realizirati poljubno logično funkcijo:

DNO in sorodne oblike:

- AND/OR
- NAND/NAND
- NOR/OR
- OR/NAND

KNO in sorodne oblike:

- OR/AND
- NOR/NOR
- NAND/AND
- AND/NOR

Primer 23: Pretvorba iz DNO (AND/OR) v NOR/OR

$$\begin{aligned} f^5 &= x_1x_2 \vee x_3x_4 \vee x_5 \\ &= \overline{\overline{x_1x_2}} \vee \overline{\overline{x_3x_4}} \vee \overline{\overline{x_5}} \\ &= \overline{\overline{x}_1 \bigvee \overline{x}_2} \vee \overline{\overline{x}_3 \bigvee \overline{x}_4} \vee \overline{\overline{x_5}} \\ &= (\overline{x}_1 \downarrow \overline{x}_2) \bigvee (\overline{x}_3 \downarrow \overline{x}_4) \bigvee \overline{x}_5 \end{aligned}$$

Primer 24: Pretvorba iz DNO (AND/OR) v OR/NAND

$$\begin{aligned} f^5 &= x_1x_2 \vee x_3x_4 \vee x_5 \\ &= \overline{\overline{x}_1 \bigvee \overline{x}_2} \vee \overline{\overline{x}_3 \bigvee \overline{x}_4} \vee \overline{\overline{x_5}} \\ &= (\overline{x}_1 \bigvee \overline{x}_2) \uparrow (\overline{x}_3 \bigvee \overline{x}_4) \uparrow \overline{x}_5 \end{aligned}$$

6.3.1 Realizacija preklopnih funkcij z operatorji Sheffer in Pierce

Največkrat preklopno funkcijo realiziramo s sistemom ($\uparrow$) ali s sistemom ($\downarrow$). Pri tem velja, da je prevedba AND/OR $\Rightarrow$ NAND/NAND enostavna in ekvivalentna prevedbi OR/AND $\Rightarrow$ NOR/NOR. Ostali prevedbi: OR/AND $\Rightarrow$ NAND/NAND in AND/OR $\Rightarrow$ NOR/NOR pa sta zahtevnejši, vendar tudi podobni.

- Prevedba iz disjunktivne normalne oblike (DNO) v Shefferjevo normalno obliko (SNO)
  - Zamenjajo se operatorji: $\bigvee \Rightarrow \uparrow$, $\& \Rightarrow \uparrow$
  - Spremenljivke, ki vstopajo v 2. nivo samostojno, se negira

  Primer 25:

$$\begin{aligned} f^5 &= x_1x_2 \vee x_3x_4 \vee x_5 \\ &= \overline{\overline{x_1x_2}} \vee \overline{\overline{x_3x_4}} \vee \overline{\overline{x_5}} \\ &= \overline{x_1 \uparrow x_2} \vee \overline{x_3 \uparrow x_4} \vee \overline{\overline{x_5}} \\ &= (x_1 \uparrow x_2) \uparrow (x_3 \uparrow x_4) \uparrow \overline{x}_5 \end{aligned}$$

- Prevedba iz disjunktivne normalne oblike (DNO) v zapis s Pierceovimi operatorji
  - Zamenjajo se operatorji: $\vee \Rightarrow \downarrow$, $\& \Rightarrow \downarrow$
  - Vse spremenljivke, razen tistih, ki vstopajo v 2. nivo samostojno, se negira
  - Negira se funkcija

Primer 26:

$$\begin{aligned} f^5 &= x_1x_2 \vee x_3x_4 \vee x_5 \\ &= (\overline{x}_1 \downarrow \overline{x}_2) \vee (\overline{x}_3 \downarrow \overline{x}_4) \vee x_5 \\ &= \overline{(\overline{x}_1 \downarrow \overline{x}_2) \downarrow (\overline{x}_3 \downarrow \overline{x}_4) \downarrow x_5} \end{aligned}$$

- Prevedba iz konjunktivne normalne oblike (KNO) v Pierceovo normalno obliko (PNO)
  - Zamenjajo se operatorji: $\vee \Rightarrow \downarrow$, $\& \Rightarrow \downarrow$
  - Spremenljivke, ki vstopajo v 2. nivo samostojno, se negira

Primer 27:

$$\begin{aligned} f^5 &= (x_1 \vee x_2) \cdot (x_3 \vee x_4) \cdot x_5 \\ &= \overline{\overline{x_1 \vee x_2}} \cdot \overline{\overline{x_3 \vee x_4}} \cdot \overline{\overline{x_5}} \\ &= \overline{x_1 \downarrow x_2} \cdot \overline{x_3 \downarrow x_4} \cdot \overline{\overline{x_5}} \\ &= (x_1 \downarrow x_2) \downarrow (x_3 \downarrow x_4) \downarrow \overline{x}_5 \end{aligned}$$

- Prevedba iz konjunktivne normalne oblike (KNO) v zapis s Shefferjevimi operatorji
  - Zamenjajo se operatorji: $\vee \Rightarrow \downarrow$, $\& \Rightarrow \downarrow$
  - Vse spremenljivke, razen tistih, ki vstopajo v 2. nivo samostojno, se negira
  - Negira se funkcija

Primer 28:

$$\begin{aligned} f^5 &= (x_1 \vee x_2) \cdot (x_3 \vee x_4) \cdot x_5 \\ &= (\overline{x}_1 \uparrow \overline{x}_2) \cdot (\overline{x}_3 \uparrow \overline{x}_4) \cdot x_5 \\ &= \overline{(\overline{x}_1 \uparrow \overline{x}_2) \uparrow (\overline{x}_3 \uparrow \overline{x}_4) \uparrow x_5} \end{aligned}$$

6.3.2 Reed-Müllerjeva oblika

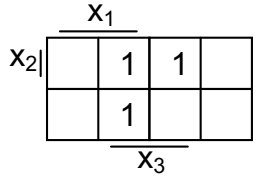
Tudi ta oblika je univerzalna, saj z njo opišemo poljubno logično funkcijo. Zasnovana je na funkcijsko polnem sistemu ($\&$, $\triangledown$, 1). Reed Müllerjevo obliko preklopne funkcije podaja izraz

$$\begin{aligned} f^{(n)} = {} & A_0 \triangledown A_1 \cdot x_1 \triangledown A_2 \cdot x_2 \triangledown \ldots \triangledown A_n \cdot x_n \\ & \triangledown A_{n+1} \cdot x_1 \cdot x_2 \triangledown A_{n+2} \cdot x_1 \cdot x_3 \triangledown \ldots \\ & \vdots \\ & \triangledown A_{2^n-1} \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_n \end{aligned}$$

kjer so A_i preklopne konstante (0 ali 1), ki jih moramo poiskati za vsako preklopno funkcijo posebej na naslednji način:

- preklopno funkcijo zapišemo v PDNO,
- operatorje $\vee$ zamenjamo z operatorji $\triangledown$,
- negacije spremenljivk $\overline{x}_i$ zamenjamo z $(1 \triangledown x_i)$,
- opravimo oklepaje, na primer, $x_i(1 \triangledown x_j) = x_i \triangledown x_i x_j$ in nazadnje
- pare enakih izrazov črtamo, saj velja $x \triangledown x = 0$.

Primer 29: Zapišimo preklopno funkcijo $f^{(3)} = \vee(3, 5, 7)$ v Reed-Müllerjevo obliko.



$$\begin{aligned} f^{(3)} &= \overline{x}_1 x_2 x_3 \vee x_1 \overline{x}_2 x_3 \vee x_1 x_2 x_3 \\ &= \overline{x}_1 x_2 x_3 \triangledown x_1 \overline{x}_2 x_3 \triangledown x_1 x_2 x_3 \\ &= (1 \triangledown x_1) x_2 x_3 \triangledown x_1 (1 \triangledown x_2) x_3 \triangledown x_1 x_2 x_3 \\ &= x_2 x_3 \triangledown x_1 x_2 x_3 \triangledown x_1 x_2 x_3 \triangledown x_1 x_3 \triangledown x_1 x_2 x_3 \\ &= x_1 x_3 \triangledown x_2 x_3 \triangledown x_1 x_2 x_3 \end{aligned}$$

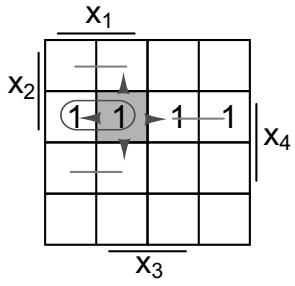
Poglavje 7

Minimizacija preklopnih funkcij

Osnova minimizacije preklopnih funkcij je v teoremu

$$x_i x_j \vee x_i \overline{x}_j = x_i \underbrace{(x_j \vee \overline{x}_j)}_{1} = x_i \quad .$$

Dva sosednja konjunktivna izraza, ki vključujeta n spremenljivk in sta disjunktivno povezana, se skrajšata v konjunkcijo dolžine $n-1$.
Spodnja slika prikazuje idejo minimizacije v Veitchovem diagramu. Označeni minterm lahko s sosednjimi mintermi (nanje kažejo puščice) tvori konjunktivne izraze. Konjunktivni izraz, ki je označen z ovalom pa lahko tvori krajše konjunktivne izraze z konjunktivnimi izrazi, ki jih označujejo s črto povezani mintermi.



Ogledali si bomo dve metodi minimizacije:

- tabelarično - Quineovo metodo in
- grafično metodo z Veitchovim diagramom.

7.1 Quineova metoda minimizacije

Postopek:

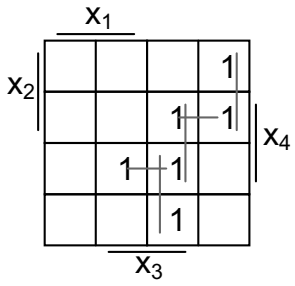
1. Funkcijo zapišemo v PDNO.
2. Poiščemo glavne vsebovalnike (konjunktivne izraze brez sosedov).
3. Izdelamo tabelo pokritij (glavni vsebovalniki/mintermi), kjer določimo pokritja mintermov s strani glavnih vsebovalnikov.
4. Poiščemo minimalno pokritje mintermov.

Primer 30: Poiščimo MNO preklopne funkcije $f^{(4)} = \bigvee(2, 3, 4, 5, 7, 11)$

1. Funkcija je že podana v PDNO.

2. V levi stolpec tabele zapišemo vse minterme, ki nastopajo v funkciji. Če se dva minterma razlikujeta samo v eni spremenljivki (x_i in $\overline{x}_i$) ju prečrtamo in v desni stolpec zapišemo konjunktivni izraz brez tiste spremenljivke. Pregledati moramo vse pare izrazov v prvem stolpcu. Ko pregledamo prvi stolpec nadaljujemo v sosednjem. Pri pregledovanju prečrtanih izrazov ne izpuščamo! Postopek je končan, ko ne najdemo nobenega novega konjunktivnega izraza. Vsi neprečrtani izrazi so glavni vsebovalniki.

$n = 4$	$n - 1 = 3$	$n - 2 = 2$
$m_2 = \overline{x}_1\overline{x}_2x_3\overline{x}_4$ $\times$	$\overline{x}_1\overline{x}_2x_3$	–
$m_3 = \overline{x}_1\overline{x}_2x_3x_4$ $\times$	$\overline{x}_1x_3x_4$	
$m_4 = \overline{x}_1x_2\overline{x}_3\overline{x}_4$ $\times$	$\overline{x}_2x_3x_4$	
$m_5 = \overline{x}_1x_2\overline{x}_3x_4$ $\times$	$\overline{x}_1x_2\overline{x}_3$	
$m_7 = \overline{x}_1x_2x_3x_4$ $\times$	$\overline{x}_1x_2x_4$	
$m_{11} = x_1\overline{x}_2x_3x_4$ $\times$		



Pari mintermov, ki tvorijo glavne vsebovalnike so grafično predstavljeni na zgornjem Veitchovem diagramu.

3. Iz tabele pokritij izberemo glavne vsebovalnike, ki edini pokrijejo enega ali več mintermov. Ti so označeni s puščico na desni strani spodnje tabele.

	m_2	m_3	m_4	m_5	m_7	m_{11}	
$\overline{x}_1\overline{x}_2x_3$	$\surd$	$\surd$					$\Leftarrow$
$\overline{x}_1x_3x_4$		$\surd$			$\surd$		
$\overline{x}_2x_3x_4$		$\surd$				$\surd$	$\Leftarrow$
$\overline{x}_1x_2\overline{x}_3$			$\surd$	$\surd$			$\Leftarrow$
$\overline{x}_1x_2x_4$				$\surd$	$\surd$		

4. Izmed dveh glavnih vsebovalnikov, ki pokrivata m_7, izberemo ugodnejšega – to je tistega, ki ima manj spremenljivk oziroma negacij.

	m_7
$\overline{x}_1x_3x_4$	$\surd$
$\overline{x}_1x_2x_4$	$\surd$

Dobimo rezultat – funkcijo zapisano v MDNO:

$$f^{(4)}_{MDNO} = \overline{x}_1\overline{x}_2x_3 \vee \overline{x}_2x_3x_4 \vee \overline{x}_1x_2\overline{x}_3 \vee \frac{\overline{x}_1x_3x_4}{\overline{x}_1x_2x_4}$$

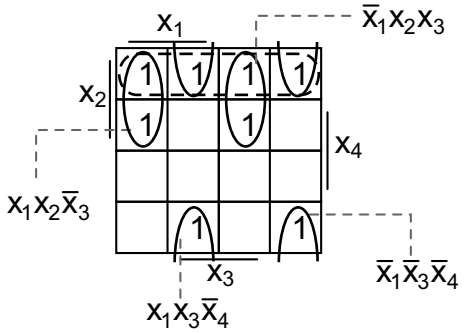
Z ulomkom smo označili, da sta obe konjunkciji popolnoma enakovredni. Pri realizaciji funkcije izberemo eno.

7.2 Grafična minimizacija z Veitchovim diagramom

Postopek:

1. Preklopno funkcijo predstavimo z Veitchovim diagramom.
2. Vizuelno poiščemo glavne vsebovalnike, oziroma konjunkcije, ki pokrivajo večje vzorce enic pravokotnih ali kvadratnih oblik. Število enic v glavnih vsebovalnikih mora biti vedno enako 2^i, $i = 1, \ldots, n$. Posamezne enice ohranimo za drugi korak.
3. Iščemo potrebne glavne vsebovalnike, to je tiste, ki najbolj ugodno pokrijejo vsaj en minterm.
4. Disjunkcija potrebnih glavnih vsebovalnikov, ki pokrijejo celotno funkcijo, je MDNO.

Primer 31: Grafična minimizacija preklopne funkcije $f^{(4)} = \bigvee(0, 4, 6, 7, 10, 12, 13, 14)$.

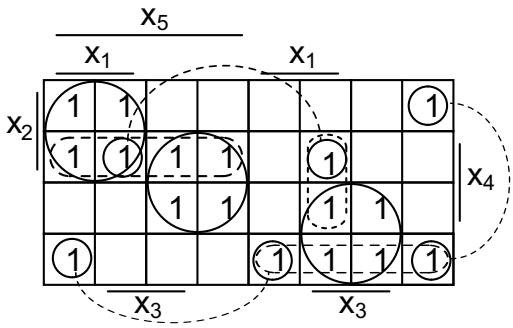


S polno črto so obkroženi glavni vsebovalniki, ki so potrebni, s črtkano pa tisti, ki ni potreben. MDNO preklopne funkcije nam predstavlja disjunkcija potrebnih glavnih vsebovalnikov

$$f^{(4)}_{MDNO} = \overline{x}_1 x_2 x_3 \vee \overline{x}_1 \overline{x}_3 \overline{x}_4 \vee x_1 x_2 \overline{x}_3 \vee x_1 x_3 \overline{x}_4$$

Primer 32: Grafična minimizacija preklopne funkcije $f^{(5)} = \bigvee(0, 3, 4, 6, 7, 8, 11, 15, 16, 17, 20, 22, 25, 27, 29, 30, 31)$.

- Funkcijo narišemo v Veitchov diagram



- Najprej obkrožimo največje glavne vsebovalnike (3 kvadrate in 2 pravokotnika). Trije mintermi ostanejo neopredeljeni.
- Poiščemo potrebne glavne vsebovalnike – na sliki so označeni s polnimi črtami.
- Zapišemo preklopno funkcijo v MDNO:

$$f^5_{MDNO} = x_1 x_2 x_5 \vee \overline{x}_1 x_4 x_5 \vee \overline{x}_2 x_3 \overline{x}_5 \vee \overline{x}_1 \overline{x}_3 \overline{x}_4 \overline{x}_5 \vee x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \vee \frac{x_1 x_2 x_3 x_4}{x_1 x_3 x_4 \overline{x}_5}$$

7.3 Minimalna normalna oblika preklopne funkcije – MNO

Preklopno funkcijo zapisano v PDNO ali v kateri drugi DNO lahko vedno zapišemo v minimalni disjunktivni normalni obliki (MDNO). Enako velja za konjunktivne normalne oblike preklopnih funkcij, ki jih lahko skrčimo v minimalno konjunktivno normalno obliko (MKNO). MDNO ali MKNO oblika preklopne funkcije, ki je ugodnejša za realizacijo se imenuje minimalna normalna oblika (MNO).

Za realizacijo je ugodnejša tista oblika, ki ima manjše število operatorjev oziroma manjše število vhodov pri enakem številu operatorjev.

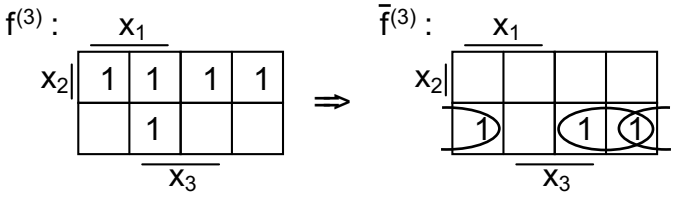
Določanje MKNO preklopne funkcije

Postopek je naslednji:

- Funkcijo, podano v PDNO, negiramo: $f \Rightarrow \overline{f}$.
- Negirani funkciji poiščemo MDNO: $\overline{f}_{MDNO}$.
- Z negacijo f_{MDNO} in uporabo de Morganovih izrekov dobimo MKNO: $\overline{\overline{f}_{MDNO}} \Rightarrow f_{MKNO}$.

Primer 33: Za preklopno funkcijo $f^{(3)} = \bigvee(2,3,5,6,7)$ določimo MKNO

Funkcijo predstavimo z Veitchovim diagramom. S komplementiranjem Veitchovega diagrama dobimo negirano funkcijo.

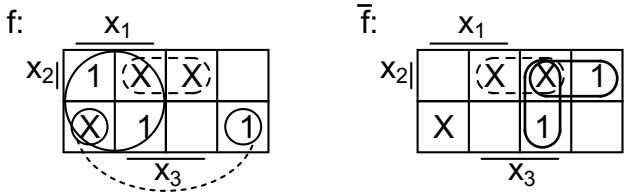


$$\begin{aligned}
\overline{f}_{MDNO} &= \overline{x}_1\overline{x}_2 \vee \overline{x}_2\overline{x}_3 \\
\overline{\overline{f}_{MDNO}} &= \overline{\overline{x}_1\overline{x}_2 \vee \overline{x}_2\overline{x}_3} = (x_1 \vee x_2) \cdot (x_2 \vee x_3) \\
f_{MKNO} &= (x_1 \vee x_2) \cdot (x_2 \vee x_3)
\end{aligned}$$

7.4 Minimizacija funkcij z redundancami

Včasih določene kombinacije vhodnih spremenljivk niso mogoče, ali pa jih ne nameravamo uporabiti. Takšnim kombinacijam pravimo redundantne. Označimo jih z $\times$ ali d (*ang.* don't care). Pri minimizaciji lahko vsako redundanco posebej obravnavamo kot logično 0 ali pa kot logično 1, odvisno od tega, kaj je bolje z vidika minimizacije.

Primer 34: Poiščimo MNO za preklopno funkcijo z redundancami
$f(x_1, x_2, x_3) = \bigvee(0, 5, 6); \bigvee_X(3, 4, 7)$



$$f_{MDNO} = x_1 \vee \overline{x}_2\overline{x}_3$$

$$\overline{\overline{f}_{MDNO}} = \overline{\overline{x}_1 x_2 \vee \overline{x}_1 x_3}$$
$$f_{MKNO} = (x_1 \vee \overline{x}_2) \cdot (x_1 \vee \overline{x}_3)$$

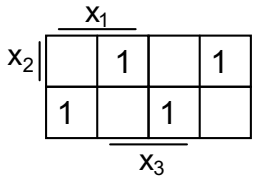
Funkcija f_{MDNO} ima en dvovhodni operator konjunkcije na prvem nivoju in en dvovhodni operator disjunkcije na drugem nivoju – skupaj torej 2 operatorja in 4 vhode. Podobno ima f_{MKNO} na prvem nivoju dva dvovhodna operatorja disjunkcije in en dvovhodni operator konjunkcije na drugem nivoju – skupaj 3 operatorje in 6 vhodov. Ker je f_{MDNO} ugodnejša s stališča realizacije, velja

$$f_{MNO} = f_{MDNO} = x_1 \vee \overline{x}_2\overline{x}_3 \quad .$$

Poglavje 8

Simetrične funkcije

Nekatere funkcije so izrazito neugodne z vidika minimizacije, ker njihovi mintermi ležijo pretežno na diagonalnih povezavah, kar onemogoča združevanje na osnovi sosednosti (spodnja slika)



Omenjene lastnosti imajo tudi simetrične funkcije. Funkcija je simetrična, če ima pri vseh kombinacijah vhodnih spremenljivk z istim številom enic vrednost 1. Simetrične funkcije lahko zato opisujemo s posebnim zapisom

$$f^{(n)} = f_A(x_1^{\omega_1}, x_2^{\omega_2}, \ldots, x_n^{\omega_n}) \quad ,$$

kjer je

- A simetrijska množica, ki lahko vsebuje poljubno podmnožico množice $(0, 1, 2, \ldots, n)$ in določa število vhodnih spremenljivk, ki imajo vrednost 1,
- $(x_1^{\omega_1}, x_2^{\omega_2}, \ldots, x_n^{\omega_n})$ pa je simetrijski nabor, ki določa literale posameznih spremenljivk (x_i ali $\overline{x}_i$)

$$x_i^{w_i} = \begin{cases} x_i & \text{za} \quad \omega_i = 1 \\ \overline{x_i} & \text{za} \quad \omega_i = 0 \end{cases}$$

8.1 Ugotavljanje simetričnosti preklopne funkcije

Metodo bomo ilustrirali kar na primeru.

Primer 35: Poglejmo ali je preklopna funkcija $f^{(4)} = \bigvee(1, 2, 4, 7, 8, 13, 14)$ simetrična.

1. Vse minterme, pri katerih ima funkcija vrednost 1 zapišemo v numerični obliki in na dnu stolpcev za vsako spremenljivko dodamo razmerje ničel in enic.

	x_1	x_2	x_3	x_4
m_1	0	0	0	1
m_2	0	0	1	0
m_4	0	1	0	0
m_7	0	1	1	1
m_8	1	0	0	0
m_{13}	1	1	0	1
m_{14}	1	1	1	0
	4/3	3/4	4/3	4/3

2. Ugotovimo, ali je izpolnjen potreben pogoj za simetričnost funkcije, to je, da so razmerja pri vseh spremenljivkah enaka ali recipročno enaka.
   Za obravnavani primer je ta pogoj izpolnjen.

3. Določene spremenljivke negiramo tako, da izenačimo vsa razmerja. To je mogoče narediti na več načinov - izberemo tistega, ki je enostavnejši.
   V obravnavanem primeru je smiselno negirati spremenljivko x_2, ki ima edina recipročno razmerje ničel in enic.

4. V tabeli dodamo še stolpec U, v katerem je zapisano število enic v vsaki vrstici.

	x_1	$\overline{x}_2$	x_3	x_4	U
m_1	0	1	0	1	2
m_2	0	1	1	0	2
m_4	0	0	0	0	0
m_7	0	0	1	1	2
m_8	1	1	0	0	2
m_{13}	1	0	0	1	2
m_{14}	1	0	1	0	2
	4/3	4/3	4/3	4/3	

   Zadosten pogoj, da je funkcija simetrična, lahko sedaj vežemo na vektor U. Zadosten pogoj za simetričnost preklopne funkcije je izpolnjen, če se število enakih komponent v vektorju U ujema z izrazom

$$n^v u = \left(\begin{array}{c} n \\ u \end{array} \right) = \frac{n!}{(n-u)!u!} \quad ,$$

   kjer je $n^v u$ število vrstic (komponent) vektorja U z vrednostjo u pri n spremenljivkah.

   Za naš primer velja:

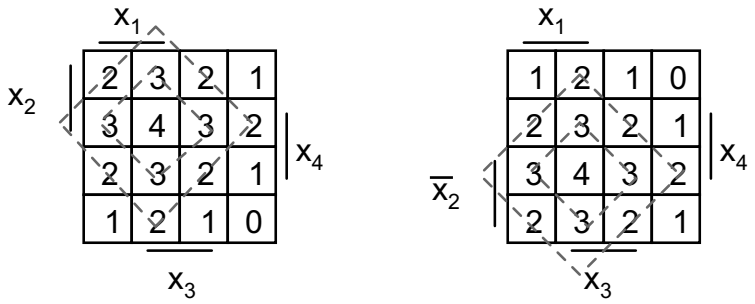
$$\begin{aligned} 4^v 2 &= \frac{4!}{(4-2)!\ 2!} = \frac{4!}{2!\ 2!} = 6 \\ 4^v 0 &= \frac{4!}{(4-0)!\ 0!} = \frac{4!}{4!\ 0!} = 1 \end{aligned}$$

   Ker imamo 6 vrstic z vrednostjo 2 in 1 vrstico z vrednostjo 0, je zadostni pogoj izpolnjen in funkcija je simetrična. Zapišemo jo lahko kot

$$f^{(4)} = f_{(0,2)}(x_1, \overline{x}_2, x_3, x_4)$$

Simetričnost preklopne funkcije lahko preverimo tudi z Veitchovim diagramom, pri katerem v kvadrate zapišemo število vhodnih spremenljivk, ki imajo vrednost 1. Funkcija je simetrična, če ima vrednost 1 v vseh kvadratih z enakim številom enic.

Primer 36: Veitchova diagrama za nabora spremenljivk (x_1, x_2, x_3, x_4) in $(x_1, \overline{x}_2, x_3, x_4)$.



Primer 37: Zapišimo simetrično preklopno funkcijo $f_{(0,2)}(x_1, \overline{x}_2, x_3, x_4)$ v Veitchov diagram.

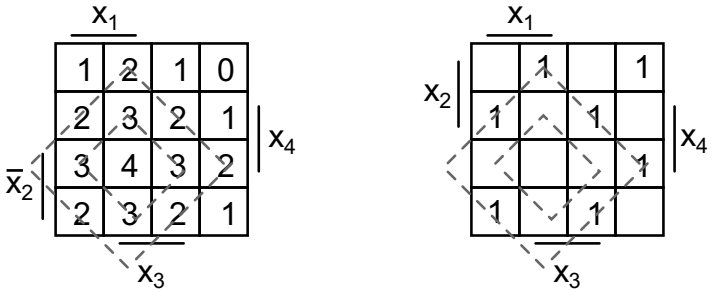
Simetrijska množica določa, da v funkciji nastopajo le tiste konjunkcije, ki imajo pri danem simetrijskem naboru 0 ali 2 spremenljivki enaki ena. V spodnji pravilnostni tabeli so izbrane vse ustrezne kombinacije spremenljivk. Tabelo preoblikujemo, da dobimo minterme, ki nastopajo v simetrični funkciji.

x_1	$\overline{x}_2$	x_3	x_4
0	0	0	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1

$\Rightarrow$

x_1	x_2	x_3	x_4	m_i
0	1	0	0	m_4
1	0	0	0	m_8
1	1	1	0	m_{14}
1	1	0	1	m_{13}
0	0	1	0	m_2
0	0	0	1	m_1
0	1	1	1	m_7

Velja $f^{(4)} = f_{(0,2)}(x_1, \overline{x}_2, x_3, x_4) = \mathsf{V}(1, 2, 4, 7, 8, 13, 14)$. Na spodnji sliki vidimo, da ima obravnavana preklopna funkcija enice res v vseh kvadratih, ki so označeni z 0 in 2.



8.2 Lastnosti simetričnih funkcij

Vzemimo simetrično funkcijo $f_A(x_1^{\omega_1}, \ldots, x_n^{\omega_n})$.

- Negacija simetrične funkcije je spet simetrična funkcija

  $$\overline{f}_A(x_1^{w_1}, \ldots, x_n^{w_n}) = f_B(x_1^{w_1}, \cdots, x_n^{w_n}) \quad ,$$

  kjer je $B = \overline{A}$ (komplement množice A)

- Funkcija z negiranim simetrijskim naborom je simetrična funkcija

  $$f_A(x_1^{\overline{w}_1}, \cdots, x_n^{\overline{w}_n}) = f_B(x_1^{w_1}, \cdots, x_n^{w_n}) \quad ,$$

  kjer za $b_i \in B$ velja $b_i = n - a_i$

- Funkcija dualna simetrični funkciji je simetrična funkcija

  $$f_A^d(x_1^{w_1}, \cdots, x_n^{w_n}) = f_B(x_1^{w_1}, \cdots, x_n^{w_n}) \quad ,$$

  kjer za $b_i \in B$ velja $b_i = n - \overline{a}_i, \overline{a}_i \in \overline{A}$

- Disjunkcija dveh simetričnih funkcij je spet simetrična funkcija

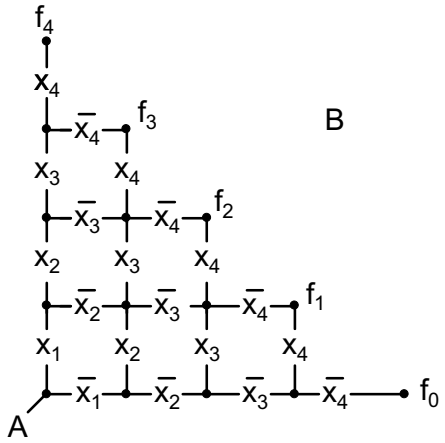
  $$f_A \vee f_B = f_C \quad , \quad C = A \cup B$$

- Konjunkcija dveh simetričnih funkcij je spet simetrična funkcija

  $$f_A \cdot f_B = f_C \quad , \quad C = A \cap B$$

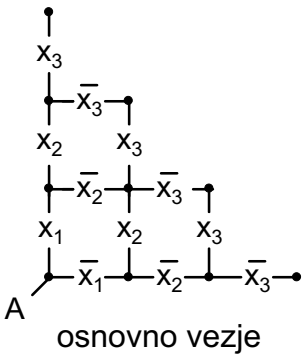
8.3 Realizacija simetričnih funkcij s Caldwellovim vezjem

Caldwellovo vezje je trikotno vezje, sestavljeno iz stikal (kontaktov), ki so pri realizaciji osnovnega nabora $(x_1, \ldots, x_n)$ v vodoravni smeri negirana, v navpični smeri pa nenegirana. V primeru spremembe nabora se ustrezno spremenijo tudi kontakti v vezju.

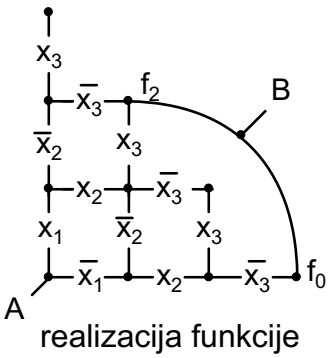


Tok steče med A in B samo, ko ima simetrična funkcija vrednost 1. Oznake f_0 do f_4 predstavljajo izhode za ustrezna simetrijska števila.

Primer 38: Realizirajmo simetrično funkcijo $f_{(0,2)}(x_1, \overline{x}_2, x_3)$ s Caldwellovim vezjem



osnovno vezje



realizacija funkcije

Poglavje 9

Verjetnostne funkcije

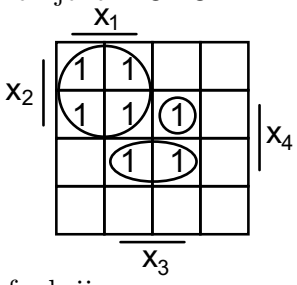
Če preklopne spremenljivke nadomestimo z verjetnostmi, da imajo spremenljivke vrednost 1, potem iz preklopne funkcije dobimo verjetnostno funkcijo. Verjetnostna funkcija nam pove verjetnost, da je vrednost funkcije 1.
Pretvorbo preklopne funkcije v verjetnostno funkcijo lahko naredimo, če preklopno funkcijo zapišemo v ortogonalni normalni obliki (ONO). Za ortogonalno normalno obliko je značilno, da se konjunktivni izrazi (vsebovalniki) ne prekrivajo. Preklopna funkcija zapisana v PDNO je tako že podana v ONO.
Postopek pretvorbe preklopne funkcije v verjetnostno:

- Funkcijo zapišemo v Veitchov diagram.
- Poiščemo minimalno ortogonalno normalno obliko (MONO), to je ONO, ki je podana s kar najmanj vsebovalniki, ki se med seboj ne prekrivajo.
- Spremenljivke x_i, ki nastopajo v preklopni funkciji nadomestimo z verjetnostmi p_i, logične operatorje pa z algebraičnimi: operator disjunkcije zamenjamo z vsoto ($\vee \Rightarrow +$), operator konjunkcije s produktom ($\& \Rightarrow \cdot$), negacijo $\overline{x}_i$ pa z $(1 - p_i)$.
- Izraz ustrezno uredimo (odpravimo oklepaje, okrajšamo, ...)

Primer 39: Za preklopno funkcijo $f^{(4)} = x_1x_2 \vee x_3x_4 = \vee(3, 7, 11, 12, 13, 14, 15)$ poiščimo verjetnostno funkcijo.

Funkcijo zapišemo v Veitchov diagram in poiščemo njeno MONO

$$f^{(4)}_{MONO} = x_1x_2 \vee \overline{x}_2x_3x_4 \vee \overline{x}_1x_2x_3x_4$$



Z omenjenimi zamenjavami dobimo verjetnostno funkcijo

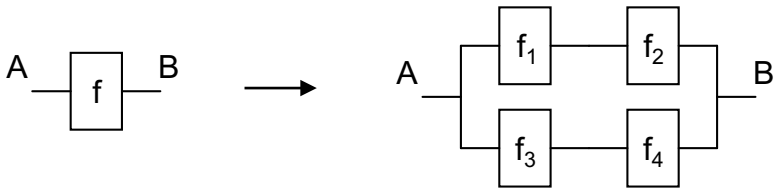
$$\mathcal{P}(f) = p_1p_2 + (1 - p_2)p_3p_4 + (1 - p_1)p_2p_3p_4$$

Zaradi lažjega računanja vzemimo $p_1 = p_2 = p_3 = p_4 = p$. Sledi

$$\begin{aligned} \mathcal{P}(f) &= p^2 + (1 - p)p^2 + (1 - p)p^3 \\ &= 2p^2 - p^4 \end{aligned}$$

9.1 Povečevanje zanesljivosti delovanja logičnih sistemov

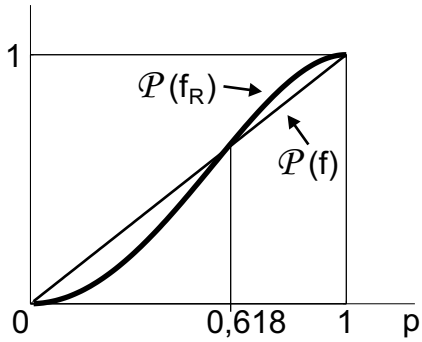
Verjetnostne funkcije uporabljamo pri računanju zanesljivosti delovanja logičnih sistemov. Logični sistemi lahko zaradi napak (kratki stik, pokvarjena stikala, ...) odpovejo. Kadar je zanesljivostna funkcija pravilnega delovanja sistema $\mathcal{P}(f)$ (ali verjetnost) premajhna za določeno aplikacijo, jo lahko povečamo s paralelno serijsko vezavo.



$\mathcal{P}$(osnovni sistem) $<$ $\mathcal{P}$(redundančni sistem)

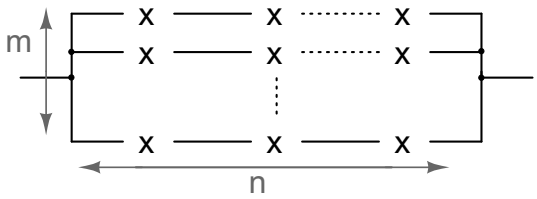
V nadaljevanju bomo privzeli, da imajo vsi dodani sistemi enako zanesljivost delovanja kot osnovi sistem, $\mathcal{P}(f) = p$.

Primer 40: Povečevanje zanesljivosti stikala
V tem primeru so funkcije enake spremenljivkam: za osnovni sistem velja $f = x$, za redundančni sistem pa $f_i = x_i$, $i = 1, \ldots, 4$. Redundančni sistem lahko v tem primeru predstavimo z logično funkcijo $f_R = x_1x_2 \vee x_3x_4$. Za to funkcijo smo že v prejšnjem primeru določili verjetnost, $\mathcal{P}(f_R) = 2p^2 - p^4$.
Na spodnjem grafu, ki prikazuje spreminjanje zanesljivosti osnovnega in redundančnega sistema vidimo, da je v primerih, ko je zanesljivost delovanja osnovnega sistema $\mathcal{P}(f) = p > 0{,}618$, zanesljivost delovanja redundančnega sistema večja kot pri osnovnem sistemu, $\mathcal{P}(f_R) > \mathcal{P}(f)$.



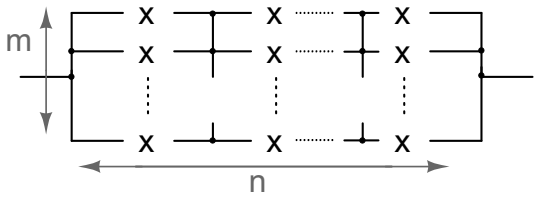
S splošnimi redundančnimi shemami lahko krivuljo za $\mathcal{P}(f_R)$ poljubno približamo zahtevam po zanesljivosti sistema:

- Serijsko-paralelna vezava



$$\mathcal{P}(f) = 1 - (1 - p^n)^m$$

- Paralelno-serijska vezava



$$\mathcal{P}(f) = (1 - (1 - p^m))^n$$

Poglavje 10

Gradniki srednje in visoke stopnje integracije

Z razvojem integriranih vezij se število tranzistorjev, ki jih vključujejo vezja neprestano povečuje. Danes integrirana vezja delimo v naslednje skupine:

- SSI (*ang.* Small Scale Integration) Gradniki nizke stopnje integracije
- MSI (*ang.* Medium Scale Integration) Gradniki srednje stopnje integracije.
- LSI (*ang.* Large Scale Integration) Gradniki visoke stopnje integracije.
- VLSI (*ang.* Very Large Scale Integration)
- ULSI (*ang.* Ultra Large Scale Integration)
- WSI (*ang.* Waffer Scale Integration)

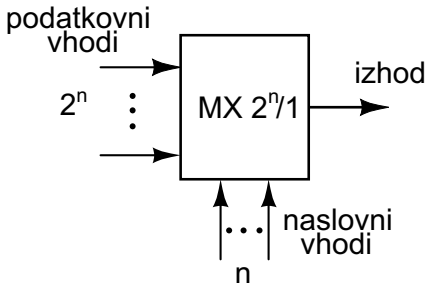
Medtem ko imajo gradniki nizke stopnje integracije (SSI) le nekaj deset tranzistorjev, imajo gradniki zelo visoke stopnje integracije (VLSI) že nekaj milijonov tranzistorjev.

10.1 Gradniki srednje stopnje integracije

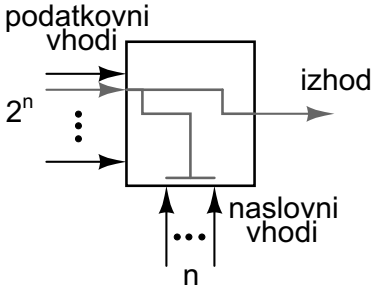
Ogledali si bomo multipleksor, demultipleksor, dekodirnik in kodirnik.

10.1.1 Multipleksor (MX)

Multipleksor ima 2 tipa vhodov: podatkovne in naslovne (adresne), ki so v medsebojni povezavi. Multipleksor z n naslovnimi vhodi ima 2^n podatkovnih vhodov in 1 izhod.



Z določitvijo vrednosti naslovnih vhodov izberemo enega od podatkovnih vhodov in njegova trenutna vrednost se prenese na izhod, kar shematično prikazuje spodnja slika.



Dobavljivi multipleksorji imajo največ 4 naslovne vhode.

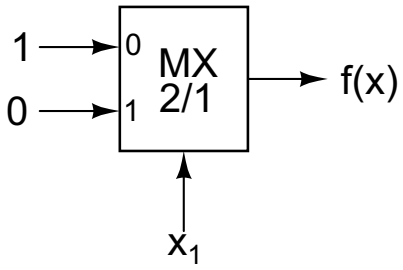
Realizacija preklopnih funkcij z multipleksorji

Z multipleksorji lahko realiziramo popolnoma splošne preklopne funkcije. Ločimo tri vrste univerzalnih rešitev:

- Trivialna rešitev: nastopa takrat, ko vse neodvisne preklopne spremenljivke pripeljemo na naslovne vhode. Tedaj na podatkovne vhode priključimo konstante (0 ali 1), ki jih dobimo iz pravilnostne tabele preklopne funkcije.

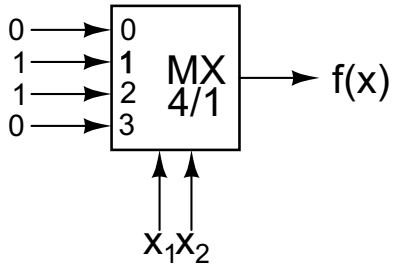
  Primer 41: Realizacija preklopne funkcije $f(x) = \overline{x}$ z multipleksorjem MX 2/1.

x	f(x)=$\overline{x}$
0	1
1	0



  Primer 42: Realizacija preklopne funkcije $f(x_1, x_2) = x_1 \bigtriangledown x_2$ z multipleksorjem MX 4/1.

i	x_1	x_2	$f = x_1 \bigtriangledown x_2$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0



- Optimalna rešitev: o njej govorimo takrat, ko je število naslovnih vhodov za 1 manjše od števila vhodnih spremenljivk. Tedaj na podatkovne vhode poleg konstant 0 in 1 pripeljemo še eno od funkcij (x ali $\overline{x}$) spremenljivke, ki ni pripeljana na naslovne vhode.

Primer 43: Realizacija preklopne funkcije $f(x_1, x_2, x_3) = \overline{x}_1 x_2 \vee x_3$ z multipleksorjem MX 4/1.

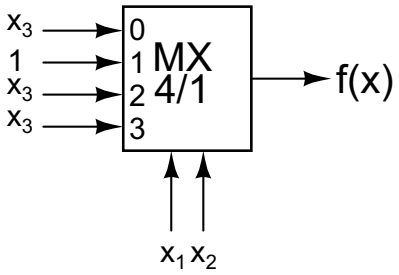
Multipleksor MX 4/1 ima dva naslovna vhoda, zato funkcijo razčlenimo po dveh spremenljivkah (poglavje 5).

$$\begin{aligned} f(x_1, x_2, x_3) &= \overline{x}_1 x_2 \vee x_3 \\ &= \overline{x}_1\overline{x}_2 f(0,0,x_3) \vee \overline{x}_1 x_2 f(0,1,x_3) \vee x_1\overline{x}_2 f(1,0,x_3) \vee x_1 x_2 f(1,1,x_3) \\ &= \overline{x}_1\overline{x}_2(x_3) \vee \overline{x}_1 x_2(1) \vee x_1\overline{x}_2(x_3) \vee x_1 x_2(x_3) \end{aligned}$$

Spremenljivki, po katerih smo funkcijo razčlenili pripeljemo na naslovne vhode multipleksorja, funkcijske ostanke (v oklepajih) pa zapišemo na ustrezne podatkovne vhode.

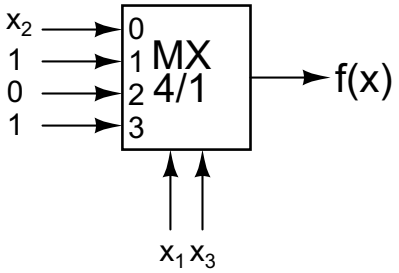
Funkcijo lahko zapišemo tudi v pravilnostno tabelo, ki jo razdelimo na 4 enake dele. Na naslovne vhode multipleksorja MX 4/1 vpišemo spremenljivki, ki sta v pravilnostni tabeli najbolj levo (x_1, x_2), na podatkovne vhode pa zapišemo funkcije (funkcijske ostanke) za vsak del tabele posebej.

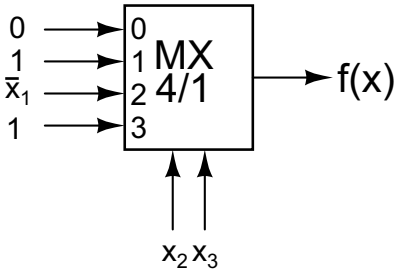
x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Namesto spremenljivk x_1, x_2 lahko uporabimo tudi spremenljivki x_1, x_3 ali pa spremenljivki x_2, x_3. Rezultat se bistveno ne spremeni.

$$\begin{aligned} f(x_1, x_2, x_3) &= \overline{x}_1 x_2 \vee x_3 \\ &= \overline{x}_1\overline{x}_3 f(0,x_2,0) \vee \overline{x}_1 x_3 f(0,x_2,1) \vee x_1\overline{x}_3 f(1,x_2,0) \vee x_1 x_3 f(1,x_2,1) \\ &= \overline{x}_1\overline{x}_3(x_2) \vee \overline{x}_1 x_3(1) \vee x_1\overline{x}_3(0) \vee x_1 x_3(1) \\ f(x_1, x_2, x_3) &= \overline{x}_1 x_2 \vee x_3 \\ &= \overline{x}_2\overline{x}_3 f(x_1,0,0) \vee \overline{x}_2 x_3 f(x_1,0,1) \vee x_2\overline{x}_3 f(x_1,1,0) \vee x_2 x_3 f(x_1,1,1) \\ &= \overline{x}_2\overline{x}_3(0) \vee \overline{x}_2 x_3(1) \vee x_2\overline{x}_3(\overline{x}_1) \vee x_2 x_3(1) \end{aligned}$$





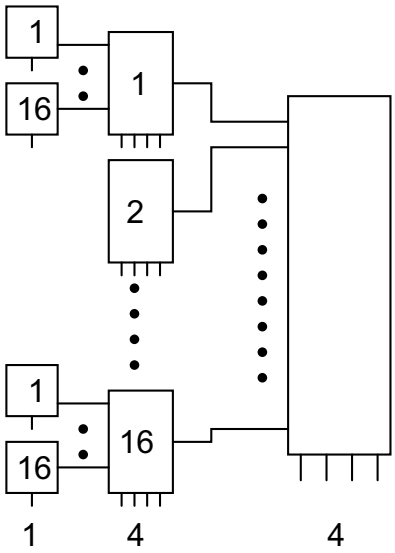
Če želimo tudi v teh dveh primerih podatkovne vhode določiti enostavno s pravilnostno tabelo, jo moramo preurediti tako, da sta spremenljivki, ki nastopata na naslovnih vhodih multipleksorja v najbolj levih stolpcih. Razlika med realizacijami je torej le v funkcijah preostale spremenljivke. Mogoče je še najslabša rešitev zadnja, s spremenljivkama x_2, x_3 na naslovnih vhodih, saj na enem od podatkovnih vhodov zahteva negator.

- Minimalna rešitev: o njej govorimo takrat, ko funkcijo realiziramo tudi z manjšimi multipleksorji oziroma s kaskadno vezavo multipleksorjev, pri kateri je število uporabljenih elementov minimalno. Seveda gre pri tem tudi za kompromis, saj moramo upoštevati tudi zakasnitev (oziroma število nivojev v kaskadi), ki naj bo čim manjša.

  Primer 44: Realizirajmo preklopno funkcijo 10 spremenljivk. Na razpolago imamo multipleksorje MX 2/1, MX 4/1, MX 8/1 in MX 16/1. Velja:

  – MX 2/1: 1 naslovni vhod, 6/čip
  – MX 4/1: 2 naslovna vhoda, 4/čip
  – MX 8/1: 3 naslovne vhode, 2/čip
  – MX 16/1: 4 naslovna vhoda, 1/čip

  Multipleksorji v istem čipu imajo iste naslovne vhode in različne podatkovne vhode ter izhode.



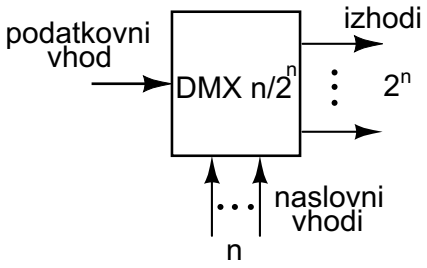
  Zgornja realizacija ima 3 zakasnilne nivoje in zahteva

  – 17 MX 16/1 kar je enakovredno 17 čipom z MX 16/1 in
  – 256 MX 2/1 oziroma 43 čipov z MX 2/1.

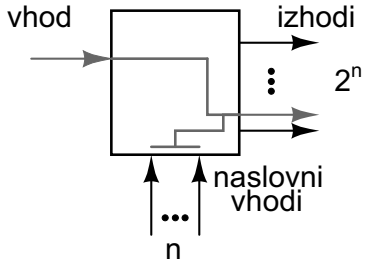
  Če bi uporabili samo MX 2/1 bi dobili kaskado z 9 nivoji, potrebovali pa bi 411 MX 2/1 oziroma 69 čipov. Ali obstaja boljša rešitev? Koliko čipov potrebujemo najmanj, če zakasnitev ni pomembna?

10.1.2 Demultipleksor (DMX)

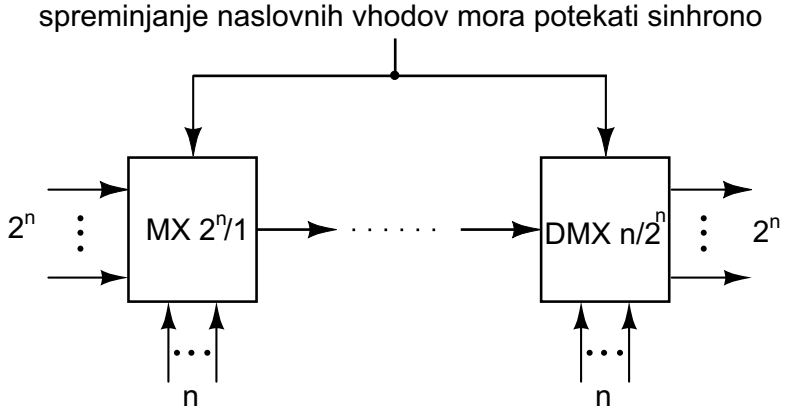
Demultipleksor DMX $n/2^n$ ima tako kot multipleksor MX $2^n/1$ n naslovnih (adresnih) vhodov. Ker je njegova funkcija obratna funkciji multipleksorja ima samo en podatkovni vhod in 2^n izhodov. Demultipleksor opisuje blok shema



Z naslovnimi vhodi določimo na katerega od izhodov se prenese vrednost, ki je na podatkovnem vhodu.

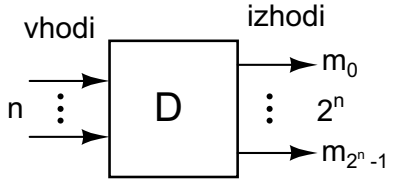


Demultipleksor se v kombinaciji z multipleksorjem uporablja pri prenosih podatkov, kjer multipleksor pretvarja paralelni podatek v serijskega, demultipleksor pa serijskega nazaj v paralelnega.



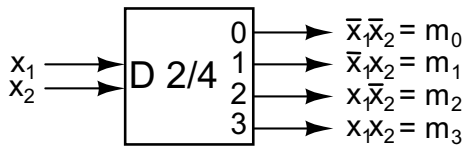
10.1.3 Dekodirnik (D)

Dekodirnik se od demultipleksorja razlikuje le po tem, da ima na podatkovnem vhodu vedno 1. Dekodirnik D $n/2^n$ podaja spodnja blok shema:



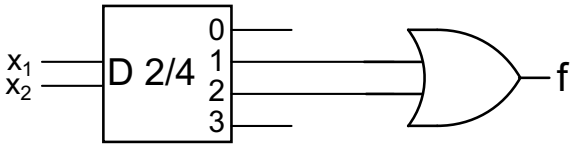
Iz pravilnostne tabele dekodirnika D 2/4

x_1	x_2	m_0	m_1	m_2	m_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

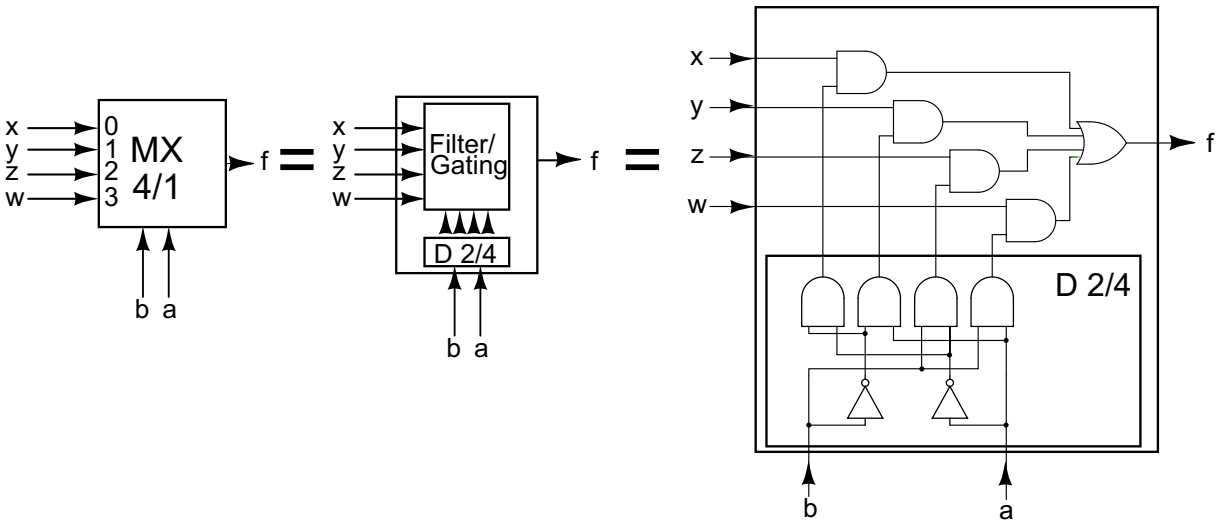


je razvidno, da opravlja funkcijo generatorja mintermov. Zato lahko z njim realiziramo poljubno funkcijo – ustrezne vhode moramo le disjunktivno povezati.

Primer 45: Preklopno funkcijo $f = x_1 \triangledown x_2 = \overline{x}_1 x_2 \vee x_1 \overline{x}_2$ realizirajmo z dekodirnikom D 2/4.

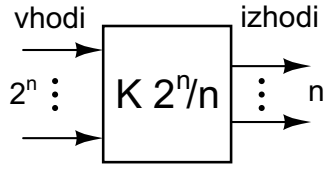


Če si podrobno ogledamo shemo multipleksorja vidimo, da dekodirnik (D) nastopa kot ena od komponent multipleksorja.



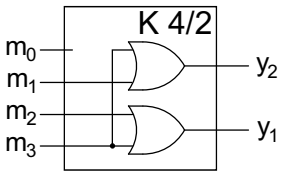
10.1.4 Kodirnik (K)

Kodirnik K $2^n/n$, ki ga prikazuje spodnja slika, je vezje z 2^n vhodi in n izhodi.



Kodirnik opravlja obratno funkcijo od dekodirnika – omogoča kompresijo vhodov, kar ponazarjata pravilnostna tabela in shema kodirnika K 4/2

m_0	m_1	m_2	m_3	y_1	y_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



10.2 Gradniki visoke stopnje integracije (LSI)

Med gradnike visoke stopnje integracije uvrščamo univerzalne logične gradnike oziroma programabilne logične naprave – PLD (*ang.* Programmable Logical Devices). Med omenjene naprave štejemo

- računalniške gradnike:
  - RAM (*ang.* Random Access Memory),
  - ROM (*ang.* Read Only Memory),
  - mikroprocesorje
- logične gradnike:
  - PLD, ki jih delimo na
    - PAL (*ang.* Programable Array Logic),
    - PLA (*ang.* Programable Logic Array) in
    - PLE (*ang.* Programable Logic Element) – sem sodita tudi RAM in ROM.
  - CPLD, kompleksne programabilne logične naprave (*ang.* Complex Programmable Logic Devices)
  - LCA (*ang.* Logic Cell Array).

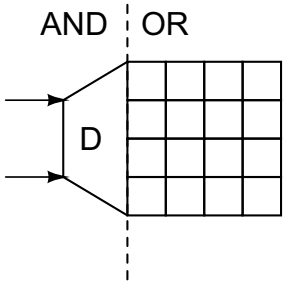
  Med tem ko sta prvi dve skupini zasnovani na tehnologiji ROM, kar pomeni, da se vsebina po izklopu napetosti ohranja, pa je zadnja skupina zasnovana na tehnologiji RAM, pri kateri se vsebina ob izklopu napetosti izgubi. Prvi dve skupini se med seboj razlikujeta predvsem po kompleksnosti celic, za zadnji dve družini pa je značilna prisotnost pomnilnih celic v vsakem gradniku.

Programirljiva vezja na osnovi tehnologije ROM se delijo tudi po načinu programiranja:

- PROM: enkratno programiranje
- EPROM: večkratno programiranje, brisanje vsebine z ultravijolično svetlobo
- EEPROM: večkratno programiranje, brisanje vsebine s povišano napetostjo

10.2.1 Gradniki PAL, PLA, PLE

Vsi gradniki v tej skupini so sestavljeni iz dveh delov: iz matrike AND, ki predstavlja konjunkcije in matrike OR, ki predstavlja disjunkcije. Vezje lahko predstavimo tudi s pravilnostno tabelo (spodaj desno)



$\dot{x}=(x_1,\ldots,x_n)$	$\dot{y}=(y_1,\ldots,y_n)$
$0\cdots0$	$\cdot$
$\vdots$	$\vdots$
$1\cdots1$	$\cdot$
$[\ AND\ ]$	$[\ OR\ ]$

Gradniki se med seboj razlikujejo predvsem po tem, kateri del vezja je spremenljiv (programirljiv). Na spodnjih slikah je programirljiv del vezja označen z $\times$:

PAL

ẋ	ẏ
×	
[*AND*]	[*OR*]

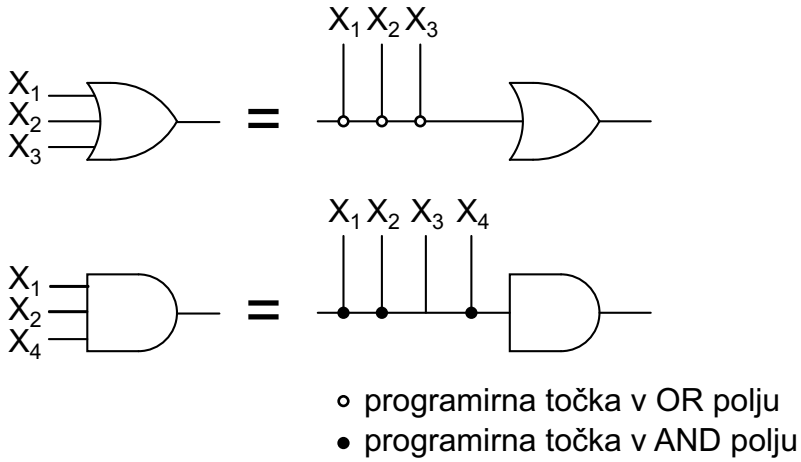
PLA

ẋ	ẏ
×	×
[*AND*]	[*OR*]

PLE

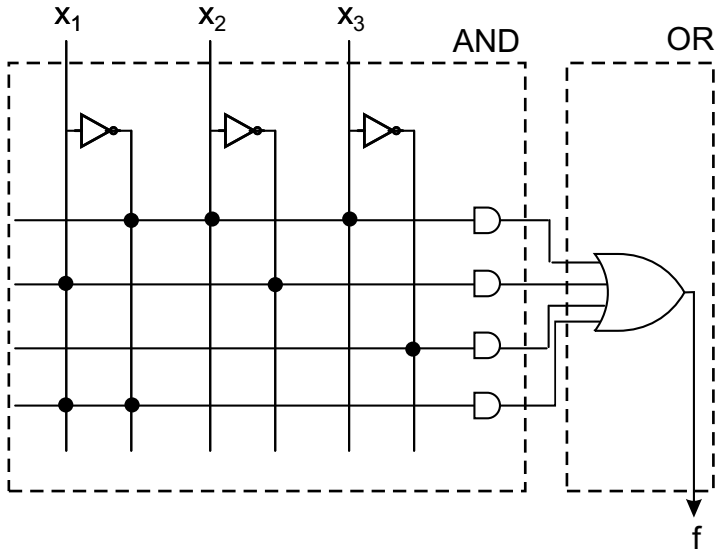
ẋ	ẏ
	×
[*AND*]	[*OR*]

Pri risanju shem za PAL, PLA in PLE gradnike si pomagamo z naslednjo poenostavitvijo pri risanju logičnih vrat:



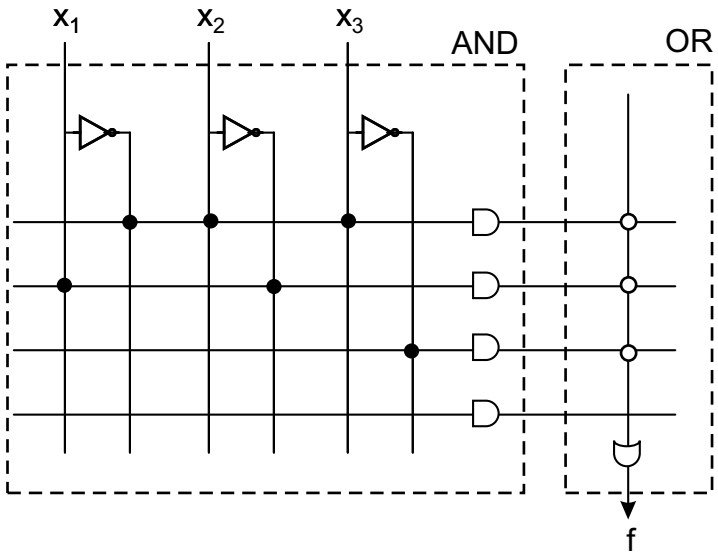
Primer 46: Ilustracija realizacije preklopne funkcije $f = \overline{x}_1 x_2 x_3 \vee x_1 \overline{x}_2 \vee \overline{x}_3$ z elementi PAL, PLA in PLE.

- PAL:



OR del vezja je nespremenljiv, zato moramo za zanesljivo delovanje vezja v AND delu vezja sprogramirati vsa AND vrata. Vrata AND, ki jih ne potrebujemo sprogramiramo tako, da je izhod vedno 0 – v zgornjem primeru smo zadnja vrata AND sprogramirali kot $x_1 \cdot \overline{x}_1 = 0$.

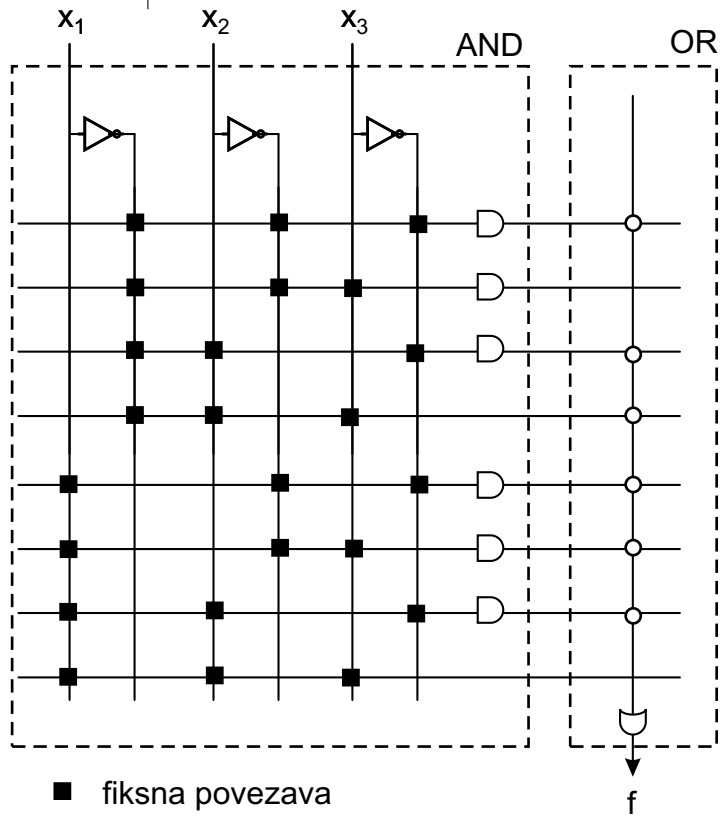
- PLA:



- PLE:
  Za PLE vezja je značilno, da je polje AND fiksno in služi kot generator mintermov. Zato poiščimo minterme naše funkcije:

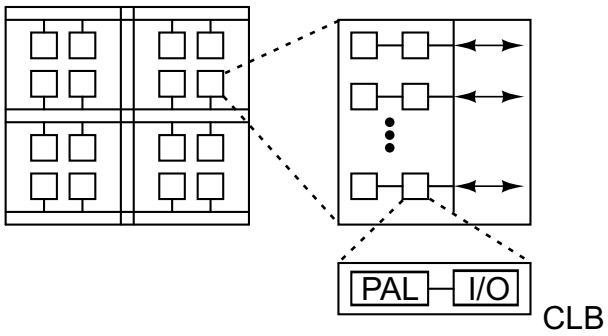
x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$\Rightarrow \qquad f = \bigvee(0,2,3,4,5,6)$$



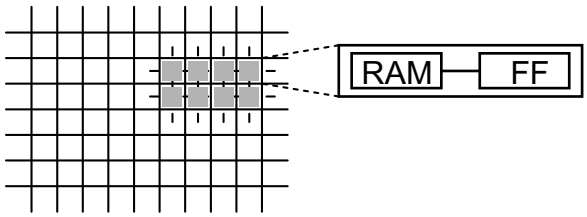
10.2.2 Gradniki CPLD

Gradniki CPLD so običajno sestavljeni iz več enakih celic. Vsaka od teh celic, imenovana spremenljiv logični blok (*ang.* CLB, Configurable Logic Block), pa vsebuje PLD gradnik in vhodno izhodni modul (I/O). Danes so v večino vhodno izhodnih modulov vključene pomnilne celice.



10.2.3 Gradniki LCA

Gradniki LCA so zasnovani na RAM tehnologiji. Za ohranitev vsebine morajo biti na vezjih,v katera so vključeni, majhne zunanje baterije in ROM elementi s pomočjo katerih se ob vsakem vklopu reinicializirajo. Zaradi zasnove na RAM tehnologiji je mogoče spreminjanje njihove vsebine v realnem času.



Vsaka celica gradnika LCA je povezana z vsemi sosednjimi celicami. Vsebuje nekaj RAM gradnikov, nekaj pomnilnih celic (FF) in nekaj multipleksorjev (MX) za programiranje poti znotraj celice.

Poglavje 11

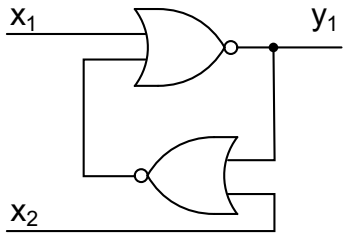
Sekvenčna logika

11.1 Pomnilne celice

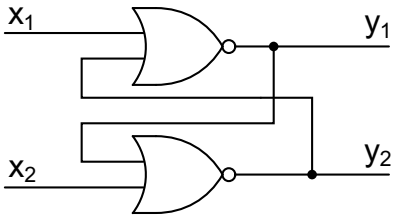
Vsaka zanka v logičnem vezju predstavlja element pomnenja. Zanimajo nas samo tisti pomnilni elementi, ki omogočajo

- kontroliran vpis vsebine – 0 ali 1 in
- branje vsebine brez njene porušitve.

Najbolj enostavno vezje, ki zadošča zgornjim zahtevam dobimo s povratno vezavo dveh operatorjev tipa NOR:



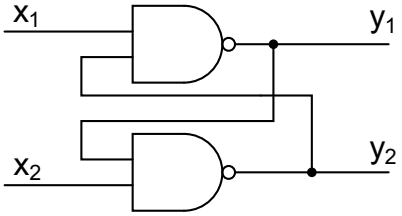
Zgornje vezje lahko narišimo malo drugače in ga analizirajmo.



Če na vhoda x_1 in x_2 postavljamo različne vrednosti 0 in 1 in izračunavamo rezultat na izhodih $y_1 = x_1 \downarrow y_2$ in $y_2 = x_2 \downarrow y_1$ dobimo naslednjo tabelo:

x_1	x_2	y_1	y_2	
0	1	1	0	
0	0	1	0	
1	0	0	1	
0	0	0	1	
1	1	0	0	prepovedana kombinacija
0	0	$\times$	$\times$	

Iz tabele je razvidno, da pri vhodih $(x_1, x_2) = (0, 0)$ vezje na izhodih ohranja prejšnje stanje. Vhodna kombinacija $(0, 1)$ postavi izhod y_1 na 1, izhod y_2 pa na 0. Obratno pa vhodna kombinacija $(1, 0)$ postavi izhod y_1 na 0, izhod y_2 pa na 1. Prehod iz vhodne kombinacije $(1, 1)$ v mirovno kombinacijo $(0, 0)$ ni možen, saj je zaradi tehnologije rezultat nepredvidljiv.
Podobno analizo lahko naredimo tudi za vezje sestavljeno iz vrat NAND:



Če na vhoda x_1 in x_2 ponovno postavljamo različne vrednosti 0 in 1 in izračunavamo rezultat na izhodih $y_1 = x_1 \uparrow y_2$ in $y_2 = x_2 \uparrow y_1$ dobimo naslednjo tabelo:

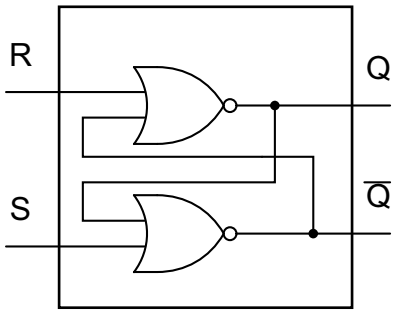
x_1	x_2	y_1	y_2	
0	1	1	0	
1	1	1	0	
1	0	0	1	
1	1	0	1	
0	0	1	1	prepovedana kombinacija
1	1	×	×	

Iz tabele je razvidno, da to vezje ohranja vrednost izhodov pri vhodni kombinaciji $(x_1, x_2) = (1, 1)$, pri vhodnih kombinacijah $(0, 1)$ in $(1, 0)$ pa se obnaša tako kot prej – pri prvi postavi y_1 na 1 in y_2 na 0, pri drugi pa y_1 na 0 in y_2 na 1. Kombinacija $(1, 1)$ je prepovedana, ker tranzistorska tehnologija ne omogoča enoumne določitve izhoda pri prehodu $(0, 0) \Rightarrow (1, 1)$.

11.1.1 Pomnilna celica RS

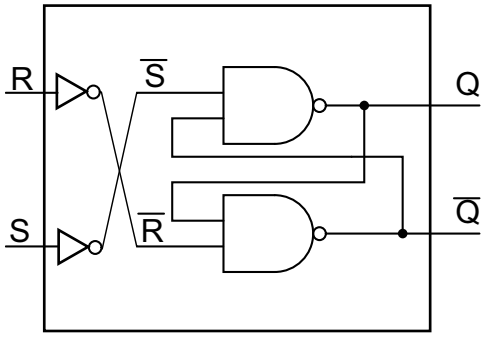
Obravnavani vezji predstavljata osnovo za pomnilno celico RS. Z izjemo prepovedanih vhodnih kombinacij za izhoda pri obeh vezjih velja $y_1 = \overline{y}_2$, zato lahko zapišemo tudi $y_1 = Q$ in $y_2 = \overline{Q}$. Ker pri vezju z NOR vrati kombinacija $x_1 = 1, x_2 = 0$ zbriše (*ang.* reset) y_1, $y_1 = 0$, kombinacija $x_1 = 0, x_2 = 1$ pa y_1 postavi (*ang.* set), $y_1 = 1$, je smiselna zamenjava oznak $x_1 \Rightarrow R$, $x_2 \Rightarrow S$.
Dobimo naslednjo karakteristično tabelo pomnilne celice RS in ustrezno shemo.

R	S	$Q(t+1)$
0	0	$Q(t)$
0	1	1
1	0	0
1	1	×



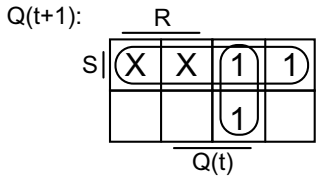
V karakteristični tabeli smo z $Q(t)$ označili trenutno vrednost izhoda Q, s $Q(t+1)$ pa novo vrednosti izhoda Q.
Če v vezju z operatorji NAND naredimo naslednje zamenjave: $y_1 \Rightarrow Q$, $y_2 \Rightarrow \overline{Q}$, $x_1 \Rightarrow \overline{S}$, $x_2 \Rightarrow \overline{R}$ dobimo enako karakteristično tabelo, le vezje je v tem primeru malo spremenjeno:

R	S	$Q(t+1)$
0	0	$Q(t)$
0	1	1
1	0	0
1	1	$\times$



Iz karakteristične tabele pomnilne celice RS

R	S	$Q(t+1)$
0	0	$Q(t)$
0	1	1
1	0	0
1	1	$\times$

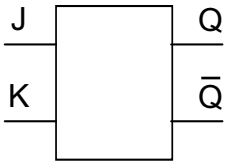


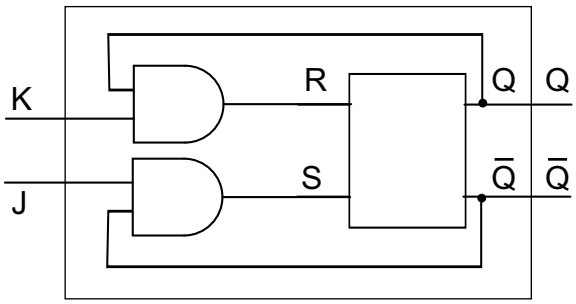
lahko zapišemo karakteristično ali pomnilno enačbo pomnilne celice RS:

$$Q(t+1) = S \vee \overline{R} \cdot Q(t) \quad , \quad R \cdot S = 0 \quad .$$

11.1.2 Pomnilna celica JK

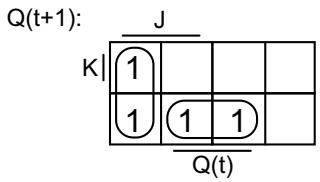
Pomnilna celica JK rešuje glavni problem pomnilne celice RS – to je prepovedano stanje pri vhodni kombinaciji $R = 1, S = 1$.





Z vezavo, dvojih dodatnih vrat AND k pomnilni celici RS, namreč odpravimo prepovedano stanje, hkrati pa dobimo še novo funkcijo – negacijo stanja pomnilne celice. To je razvidno iz delovanja pomnilne celice JK, ki ga podajata karakteristična tabela

J	K	Q(t+1)
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\overline{Q}(t)$



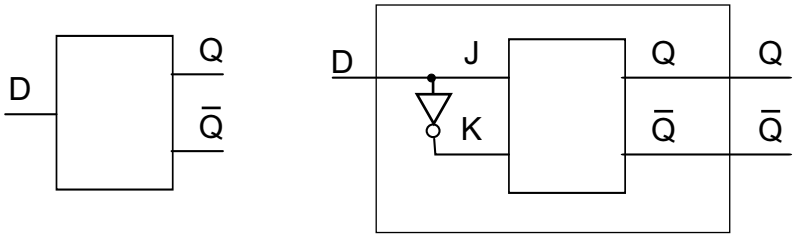
in karakteristična ali pomnilna enačba:

$$Q(t+1) = J\overline{Q}(t) \vee \overline{K}Q(t) \quad .$$

Pri pomnilni celici JK ima vhod J funkcijo postavljanja, vhod K pa funkcijo brisanja vsebine pomnilne celice.

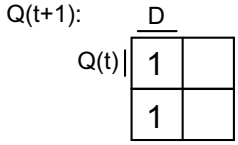
11.1.3 Pomnilna celica D

Pomnilna celica D ima samo en vhod – D (*ang.* Delay, zakasnitev). Njena naloga je, da shrani vrednost, ki jo damo na vhod. Iz pomnilne celice JK jo dobimo tako, kot prikazuje spodnja slika.



Delovanje pomnilne celice D podajata karakteristična tabela

D	$Q(t+1)$
0	0
1	1

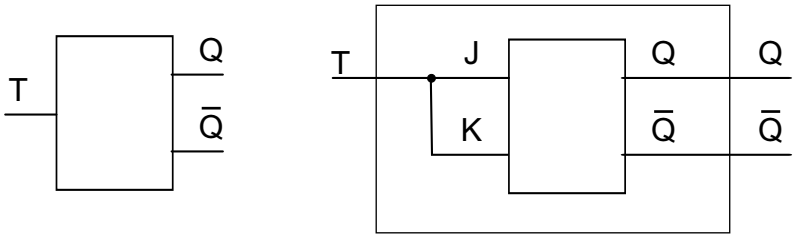


in karakteristična ali pomnilna enačba

$$Q(t+1) = D \quad .$$

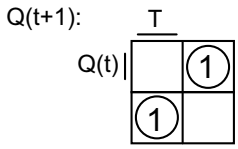
11.1.4 Pomnilna celica T

Z vhodom v pomnilno celico T določamo ali se njena vsebina ohranja ali pa spremeni. Od tod izhaja tudi njeno ime T (*ang.* Toggle, zamenjaj). Najbolj enostavno jo realiziramo s pomnilno celico JK.



Delovanje pomnilne celice T podajata karakteristična tabela

T	$Q(t+1)$
0	$Q(t)$
1	$\overline{Q}(t)$



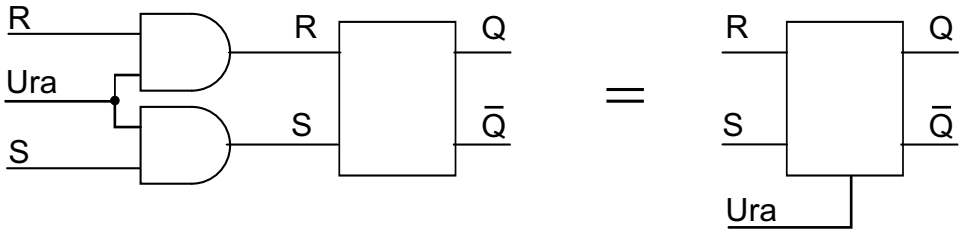
in karakteristična ali pomnilna enačba

$$\begin{aligned} Q(t+1) &= T\overline{Q}(t) \vee \overline{T}Q(t) \\ &= T \triangledown Q(t) \quad . \end{aligned}$$

11.2 Proženje pomnilnih celic

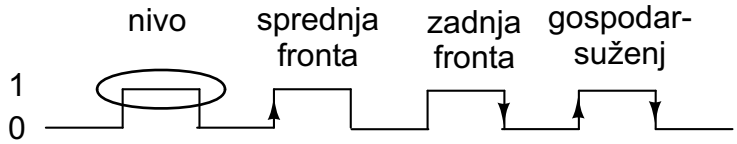
Poznamo več načinov proženja pomnilnih celic:

- asinhrono proženje: V tem primeru se notranje stanje pomnilne celice spremeni takoj ob spremembi vhoda. Spreminjanje stanj različnih pomnilnih celic v vezju ni usklajeno. Take pomnilne celice smo obravnavali do sedaj.
- sinhrono proženje: Stanja vseh pomnilnih celic v vezju se spremenijo hkrati. Ta vezja zato vsebujejo še dodaten periodični krmilni signal, časovno verigo impulzov imenovano tudi urin signal ali ura (*ang.* clock). Glede na način uporabe urinega signala ločimo:
  - proženje na nivo. Namesto izraza pomnilna celica se v tem primeru večkrat uporablja termin zapah (*ang.* latch). Vsebina pomnilne celice se spreminja ves čas, ko je urin signal v visokem stanju (zapah je odprt) in se ohranja, ko je urin signal v nizkem stanju (zapah je zaprt). Nadgradnja asinhrone pomnilne celice RS v pomnilno celico RS z nivojsko sinhronizacijo je prikazana na spodnji sliki:



  - proženje na fronto: tu se vezja delijo še po tem, na kateri prehod urinega signala reagirajo:
    - sprednja (prva) fronta: Stanje pomnilne celice se spremeni ko gre urin signal iz 0 v 1.
    - zadnja (druga) fronta: Stanje pomnilne celice se spremeni ko gre urin signal iz 1 v 0.
    - gospodar-suženj (*ang.* master-slave): Pri proženju pomnilne celice tega tipa se uporabljata sprednja in zadnja fronta.

    Načine proženja pomnilnih celic prikazuje spodnja slika.



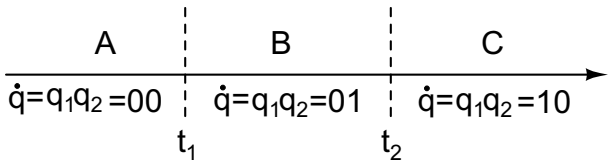
    Glede na odzivni čas in natančnost delovanja se danes največ uporabljata sinhronizaciji na sprednjo ali na zadnjo fronto urinega signala.

11.3 Sekvenčna vezja

Za sekvenčna preklopna vezja je značilno, da je naboru neodvisnih vhodnih spremenljivk dodana še časovna spremenljivka. Zato s sekvenčnim vezjem lahko realiziramo časovno odvisne (sekvenčne) probleme.
Vzemimo, da imamo sekvenčno vezje, ki ga podaja funkcija $\dot{y} = f(\dot{x}, t)$, kjer je $\dot{y} = (y_1, \ldots, y_m)$ vektor izhodnih spremenljivk in $\dot{x} = (x_1, \ldots, x_n)$ vektor vhodnih spremenljivk. Potem v časih t_1 in t_2, $t_1 \neq t_2$, v splošnem velja

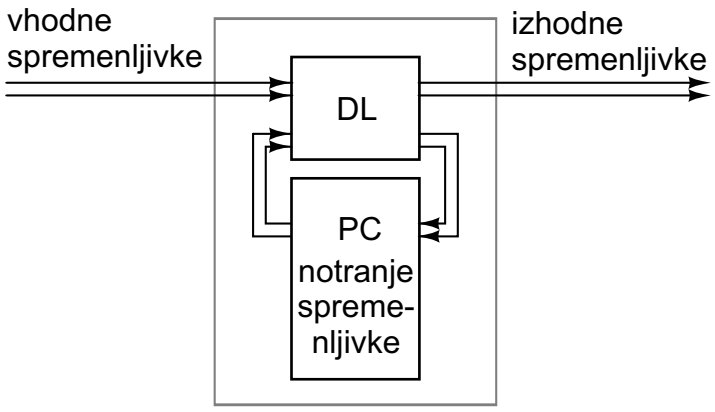
$$\begin{aligned} \dot{y}_1 &= f(\dot{x}, t_1) \\ \dot{y}_2 &= f(\dot{x}, t_2) \quad , \end{aligned}$$

pri čemer so izhodi različni, $\dot{y}_1 \neq \dot{y}_2$. Če hočemo, da je odziv vezja v različnih časih različen, mora vezje imeti sposobnost pomnenja. Povedano drugače, vezje si mora za določen čas zapomniti vrednosti izbranih (običajno notranjih) spremenljivk ($\dot{q}$).



Sekvenčno logično vezje sestavljata

- odločitveno logično vezje in
- pomnilno vezje, sestavljeno iz ene ali več pomnilnih celic.



Na shemi sekvenčnega vezja vidimo, da odločitveno vezje (decizijska logika, DL) na podlagi trenutnih vhodov in trenutne vsebine pomnilnega vezja (pomnilne celice, PC) določa izhod iz vezja in novo vsebino pomnilnega vezja. Odločitveno logično vezje v sekvenčnem vezju je zato sestavljeno iz dveh delov: iz krmilne logike, ki omogoča spreminjanje stanj (vsebine) pomnilnih celic, in iz izhodne logike.

Primer 47: Z vsemi tipi pomnilnih celic realizirajmo splošno pomnilno enačbo

$$Q(t+1) = g_1 Q(t) \vee g_2 \overline{Q}(t) \quad .$$

v kateri sta g_1 in g_2 funkciji vhodov in ostalih pomnilnih celic v vezju, ki vplivajo na stanje opazovane pomnilne celice Q.
Splošno pomnilno enačbo lahko vpišemo v pravilnostno tabelo:

g_1	g_2	Q(t)	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

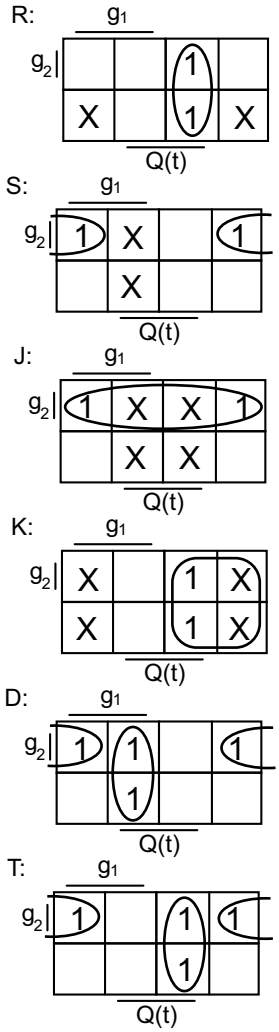
Podrobno si poglejmo pomnilno celico RS. Iz stanja $Q(t) = 0$ v stanje $Q(t+1) = 0$ lahko pridemo na dva načina: z $R = 0, S = 0$ ali pa z $R = 1, S = 0$. Iz stanja $Q(t) = 0$ v stanje $Q(t+1) = 1$ lahko pridemo samo z vhodno kombinacijo $R = 0, S = 1$. Naprej, iz stanja $Q(t) = 1$ v stanje $Q(t+1) = 0$ pridemo z $R = 1, S = 0$, iz $Q(t) = 1$ v stanje $Q(t+1) = 1$ pa pridemo z $R = 0, S = 0$ ali z $R = 0, S = 1$. Na enak način ugotovimo vhodne kombinacije tudi za vse ostale pomnilne celice. Vhodne kombinacije, potrebne za izvedbo ustreznih prehodov $Q(t) \Rightarrow Q(t+1)$, so za vse pomnilne celice navedene v spodnji tabeli prehajanja stanj:

$Q(t)$	$Q(t+1)$	R	S	J	K	D	T
0	0	×	0	0	×	0	0
0	1	0	1	1	×	1	1
1	0	1	0	×	1	0	1
1	1	0	×	×	0	1	0

Če prehod ni odvisen od določenega vhoda, je slednji označen z znakom ×.
Dopolnimo pravilnostno tabelo splošne pomnilne enačbe z vhodi v znane pomnilne celice. Vrednosti, ki jih vpišemo v ustrezne stolpce dobimo s primerjavo prehodov v pravilnostni tabeli splošne pomnilne enačbe in prehodov navedenih v tabeli prehajanja stanj.

g_1	g_2	Q(t)	Q(t+1)	R	S	J	K	D	T
0	0	0	0	×	0	0	×	0	0
0	0	1	0	1	0	×	1	0	1
0	1	0	1	0	1	1	×	1	1
0	1	1	0	1	0	×	1	0	1
1	0	0	0	×	0	0	×	0	0
1	0	1	1	0	×	×	0	1	0
1	1	0	1	0	1	1	×	1	1
1	1	1	1	0	×	×	0	1	0

Vhode v znane pomnilne celice moramo le še zapisati kot funkcijo vhodov v splošno pomnilno enačbo (g_1 in g_2) in trenutnega stanja pomnilne celice ($Q(t)$). Dobimo naslednje enačbe



$R = \overline{g}_1 Q(t)$

$S = g_2 \overline{Q}(t)$

$J = g_2$

$K = \overline{g}_1$

$D = g_1 Q(t) \vee g_2 \overline{Q}(t)$

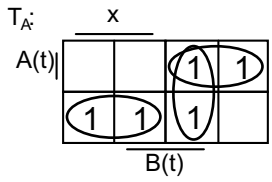
$T = \overline{g}_1 Q(t) \vee g_2 \overline{Q}(t)$

Primer 48: Relizirajmo pomnilno enačbo $A(t+1) = \overline{A}(t) \cdot B(t) \vee X$ s pomnilno celico T in multipleksorjem MX 4/1.

Pomnilna enačba opisuje dve pomnilni celici (A in B), ki imata en vhod (X). Napišimo njeno pravilnostno tabelo, nato pa z uporabo tabele prehajanja stanj v vsaki vrstici določimo ustrezen vrednost za vhod T_A v pomnilno celico A. Pri določanju vhodov smo pozorni na prehode pomnilne celice A: iz $A(t)$ v $A(t+1)$. Končna tabela ima naslednjo obliko

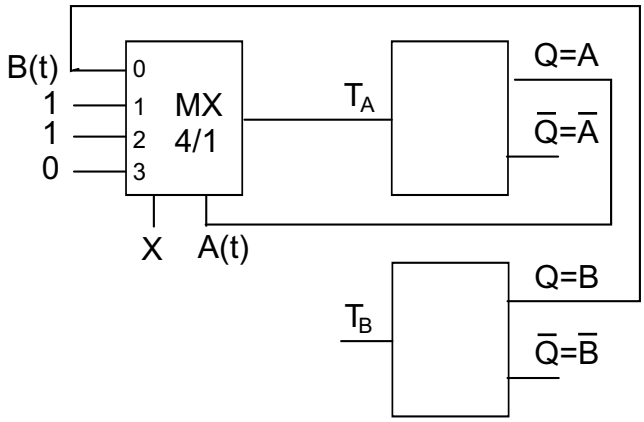
X	$A(t)$	$B(t)$	$A(t+1)$	T_A
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Vhod v pomnilno celico A še minimiziramo

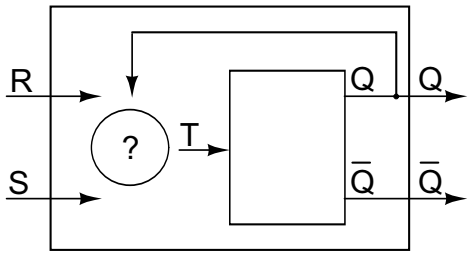


$$T_A = x \cdot \overline{A}(t) \vee \overline{x} \cdot A(t) \vee \overline{x} \cdot B(t)$$

in narišemo ustrezno shemo:



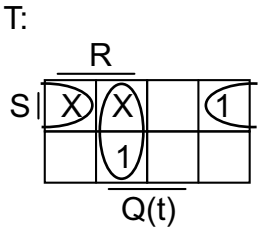
Primer 49: Pomnilno celico RS realizirajmo s pomočjo pomnilne celice T



Z uporabo karakteristične tabele pomnilne celice RS ali karakteristične enačbe napišimo pravilnostno tabelo za pomnilno celico RS in dodamo stolpec z vhodom v pomnilno celico T. Vrednost vhoda T v vsaki vrstici določimo tako, da ustreza prehodu iz $Q(t)$ v $Q(t+1)$. Pomagamo si s tabelo prehajanja stanj.

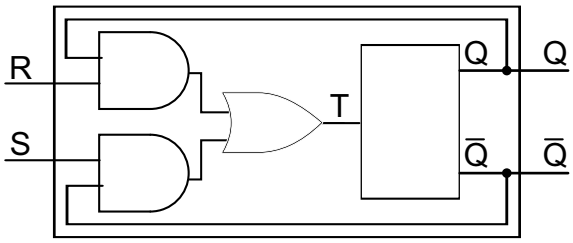
R	S	Q(t)	Q(t+1)	T
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	×	×
1	1	1	×	×

Vhod T kot funkcijo vhodov in stanja pomnilne celice RS še minimiziramo

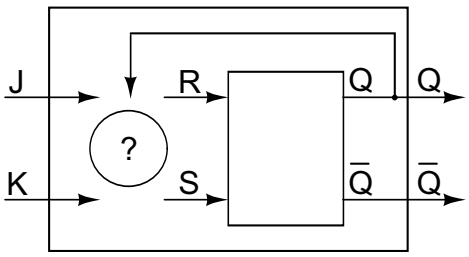


$$T = S\overline{Q}(t) \vee RQ(t)$$

in narišemo ustrezno shemo:



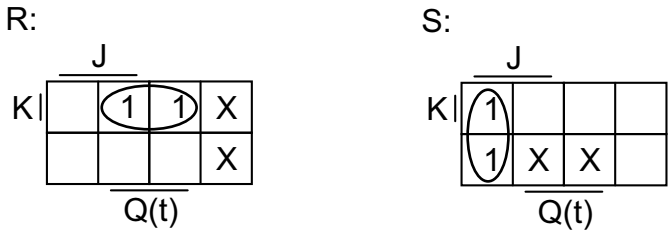
Primer 50: Pomnilno celico JK realizirajmo s pomnilno celico RS



Z uporabo karakteristične tabele pomnilne celice JK ali karakteristične enačbe napišimo pravilnostno tabelo za pomnilno celico JK in dodamo stolpca z vhodi v pomnilno celico RS. Vrednosti vhodov R in S v vsaki vrstici določimo tako, da ustrezajo prehodom iz $Q(t)$ v $Q(t+1)$. Pomagamo si s tabelo prehajanja stanj.

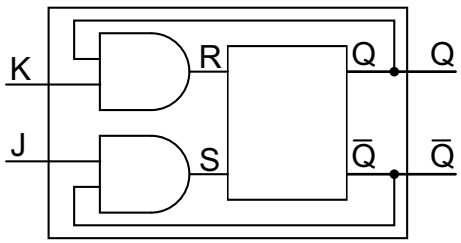
J	K	Q(t)	Q(t+1)	R	S
0	0	0	0	×	0
0	0	1	1	0	×
0	1	0	0	×	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	×
1	1	0	1	0	1
1	1	1	0	1	0

Vhoda R in S kot funkciji vhodov in stanja pomnilne celice JK še minimiziramo



$R = K \cdot Q(t)$ $S = J \cdot \overline{Q}(t)$

in narišemo ustrezno shemo:



Primer 51: S pomnilnimi celicami D izdelajmo števec po modulu 4, ki šteje takole: 0, 1, 2, 3, 0, ...

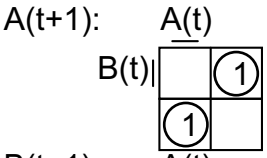
Vsaka pomnilna celica lahko zavzame samo dve stanji: 0 ali 1, zato je minimalno število pomnilnih celic potrebnih za realizacijo enako prvemu celemu številu, ki je večje ali enako $\log_2$(število stanj).
Števec pozna 4 različna števila, zato potrebujemo $\log_2 4 = 2$ pomnilni celici: A in B. Sklop dveh ali večih pomnilnih celic imenujemo register. Števila bomo kodirali binarno na naslednji način

število	A	B
0	0	0
1	0	1
2	1	0
3	1	1

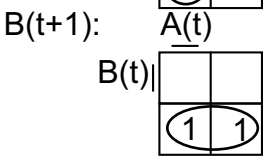
Za števec napišimo pravilnostno tabelo in s pomočjo tabele prehajanja stanj določimo vhode D_A in D_B v pomnilni celici A in B. Za določanje vhoda D_A gledamo prehod $A(t) \Rightarrow A(t+1)$, za določanje vhoda D_B pa prehod $B(t) \Rightarrow B(t+1)$.

$A(t)$	$B(t)$		$A(t+1)$	$B(t+1)$		D_A	D_B
0	0	(0)	0	1	(1)	0	1
0	1	(1)	1	0	(2)	1	0
1	0	(2)	1	1	(3)	1	1
1	1	(3)	0	0	(0)	0	0

Funkciji D_A in D_B minimiziramo:

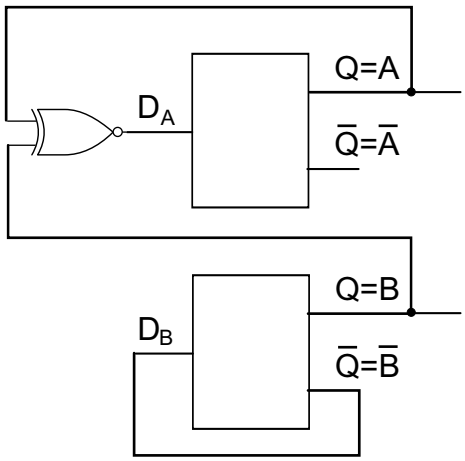


$$D_A = A(t) \cdot \overline{B}(t) \vee \overline{A}(t) \cdot B(t) = A(t) \bigtriangledown B(t)$$



$$D_B = \overline{B}(t)$$

in narišemo shemo vezja



Poglavje 12

Sekvenčni stroji

Sekvenčni stroji so matematični formalizem, ki opisuje sekvenčne preslikave. Sekvenčna preslikava je transformacija vhodov na izhod ob upoštevanju časovnih intervalov (stanj), ki vplivajo na preslikavo.
Glede na način podajanja izhodov ločimo dva tipa sekvenčnih strojev:

- sekvenčne stroje tipa Mealy in
- sekvenčne stroje tipa Moore.

Sekvenčni stroji obeh tipov se razlikujejo predvsem po funkciji za določanje izhodne črke. Medtem ko je pri sekvenčnem stroju tipa Mealy izhodna črka odvisna od trenutnega stanja in od trenutne vhodne črke je pri sekvenčnem stroju tipa Moore izhodna črka odvisna samo od trenutnega stanja.

12.1 Definicije sekvenčnih strojev

Sekvenčni stroj tipa Mealy podajamo s petorčkom $< X, Y, S, \lambda, \delta >$, sekvenčni stroj tipa Moore pa s petorčkom $< X, Y, S, \lambda, \mu >$, kjer je

- $X = \{x_1, x_2, \ldots, x_N\}$ množica vhodnih črk ali vhodna abeceda,
- $Y = \{y_1, y_2, \ldots, y_M\}$ množica izhodnih črk ali izhodna abeceda,
- $S = \{s_1, s_2, \ldots, s_K\}$ množica stanj ali notranja abeceda,
- λ operator, ki določa novo stanje, $s(t+1) = \lambda[s(t), x(t)]$,
- δ in μ pa sta operatorja, ki določata izhodno črko, $y(t) = \delta[s(t), x(t)]$ pri sekvenčnem stroju tipa Mealy oziroma $y(t) = \mu[s(t)]$ pri sekvenčnem stroju tipa Moore.

Abecede so množice črk, vsaka črka pa je kombinacija ene ali več preklopnih spremenljivk. Število črk (P) v abecedi določa minimalno število preklopnih spremenljivk (p) na osnovi izraza:

$$p = \lceil \log_2 P \rceil \quad ,$$

kjer znak $\lceil \cdot \rceil$ označuje zaokrožitev navzgor, to je k prvemu naravnemu številu, ki je večje ali enako $\log_2 P$. Na osnovi zgornjega izraza lahko izračunamo minimalno število vhodnih,

izhodnih in notranjih spremenljivk:

$$\begin{aligned} n &= \lceil \log_2 N \rceil \\ m &= \lceil \log_2 M \rceil \\ k &= \lceil \log_2 K \rceil \end{aligned}$$

V nadaljevanju bomo uporabljali naslednje oznake:

- x: vhodna črka
- y: izhodna črka
- s: notranja črka ali stanje
- ξ: vhodna spremenljivka
- η: izhodna spremenljivka
- γ: notranja spremenljivka ali vrednost pomnilne celice

12.2 Opisovanje sekvenčni strojev

Sekvenčne stroje lahko opisujemo na več načinov. Ogledali si bomo dva:

- kanonsko tabelo in
- diagram prehajanja stanj (DPS)

Glede na razlike v definiciji sta sekvenčna stroja tipa Mealy in tipa Moore različno opisana tako s kanonsko tabelo kot tudi z diagramom prehajanja stanj.

12.2.1 Opisovanje sekvenčnih strojev tipa Mealy

- Kanonska tabela: V levem stolpcu so napisana trenutna stanja, v stolpcih na desni strani pa so podana nova stanja in ustrezne izhodne črke. Stolpcev na desni je toliko kolikor je vhodnih črk (spodaj levo). Zaradi lažje berljivosti kanonsko tabelo večkrat poenostavimo tako, da vpisujemo samo indekse (spodaj desno).

	x_1	x_2	x_3
s_1	s_2/y_1	$-$	$-$
s_2	$-$	s_2/y_2	s_3/y_1
s_3	s_2/y_2	s_1/y_3	$-$

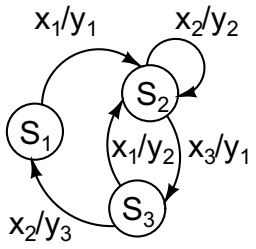
$\equiv$

	x_1	x_2	x_3
s_1	2/1	-	-
s_2	-	2/2	3/1
s_3	2/2	1/3	-

  Zadnjo vrstico zgornje kanonske tabele preberemo takole: če se sekvenčni stroj nahaja v stanju $s(t) = s_3$, potem bo v primeru, da se na vhodu pojavi črka $x(t) = x_1$ prešel v stanje $s(t+1) = s_2$, na izhodu pa se bo pojavila izhodna črka $y(t) = y_2$. V primeru, da se na vhodu pojavi črka $x(t) = x_2$, bo sekvenčni stroj prešel v stanje $s(t+1) = s_1$, pri čemer se bo na izhodu pojavila izhodna črka $y(t) = y_3$. Pri vhodni črki x_3 tako prehod kot izhodna črka nista določena vnaprej – podobno kot redundance pri preklopnih funkcijah ($- \equiv \times/\times$).

- Diagram prehajanja stanj: Sestavljen je iz vozlišč (krogov), ki predstavljajo stanja in usmerjenih povezav (puščic), ki ponazarjajo spremembo stanja. Nad puščicami sta označeni vhodna črka pri kateri pride do prehoda in ustrezna izhodna črka.

  Diagram prehajanja stanj za zgornjo kanonsko tabelo:



12.2.2 Opisovanje sekvenčnih strojev tipa Moore

- Kanonska tabela: V levem stolpcu so napisana trenutna stanja, v stolpcu na skrajni desni pa ustrezne izhodne črke. V stolpcih na sredini (pod vhodnimi črkami) so podana nova stanja. Stolpcev na sredini je toliko kolikor je vhodnih črk. Zaradi večje preglednosti lahko v kanonsko tabelo vpisujemo samo indekse stanj in izhodnih črk (spodaj desno).

	x_1	x_2	x_3	
s_1	s_2	$-$	$-$	y_1
s_2	$-$	s_2	s_3	$-$
s_3	s_2	s_1	$-$	y_2

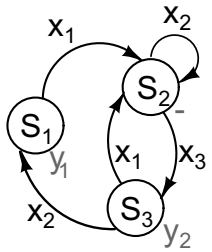
$\equiv$

	x_1	x_2	x_3	
s_1	2	-	-	1
s_2	-	2	3	-
s_3	2	1	-	2

  Zadnjo vrstico zgornje kanonske tabele preberemo takole: če se sekvenčni stroj nahaja v stanju $s(t) = s_3$, je na izhodu izhodna črka $y(t) = y_2$. V primeru, da se na vhodu pojavi črka $x(t) = x_1$ bo sekvenčni stroj prešel v stanje $s(t + 1) = s_2$. V primeru, da se na vhodu pojavi črka $x(t) = x_2$, bo sekvenčni stroj prešel v stanje s_1. Pri vhodni črki x_3 prehod ni določen vnaprej – podobno kot redundance pri preklopnih funkcijah ($- \equiv \times$). Lahko pa se zgodi (tako kot v drugi vrstici), da ni določena izhodna črka.

- Diagram prehajanja stanj: Sestavljen je iz vozlišč (krogov), ki predstavljajo stanja in usmerjenih povezav (puščic), ki ponazarjajo spremembo stanja. Ob vsakem vozlišču je podana izhodna črka (če je definirana), nad puščicami pa je označena vhodna črka pri kateri pride do prehoda.

  Diagram prehajanja stanj za zgornjo kanonsko tabelo:



12.3 Ekvivalenca sekvenčnih strojev

Sekvenčni stroji tipa Mealy so ekvivalentni sekvenčnim strojem tipa Moore – poljubno sekvenčno vezje lahko opišemo z enim ali pa z drugim sekvenčnim strojem. Zato lahko sekvenčni stroj enega tipa pretvorimo v sekvenčni stroj drugega tipa.

12.3.1 Pretvorba Mealy ⇒ Moore

Zapišimo sekvenčni stroj tipa Mealy (M) s petorčkom $< X, Y, S, \lambda, \delta >$ in sekvenčni stroj tipa Moore (M') s petorčkom $< X', Y', S', \lambda', \mu' >$. Potem velja:

- Vhodne in izhodne črke:
$$\begin{aligned} X' &= X \\ Y' &= Y \end{aligned}$$

- Stanja:
$$\begin{aligned} S' &= S_1 \cup S_2 \\ S_1 &= \{s_{km} \mid \lambda(s_i, x_j) = s_k, \delta(s_i, x_j) = y_m, s_i \in S, x_j \in X\} \\ S_2 &= \{s_k \mid \lambda(s_i, x_j) \neq s_k, s_i \in S, x_j \in X\} \end{aligned}$$

- Operator prehajanja stanj:
Če $\lambda(s_i, x_j) = s_k$ in $\delta(s_i, x_j) = y_m$ potem $\lambda'(s_{in}, x_j) = s_{km}$ za vse $s_{in} \in S_1$ in $\lambda'(s_i, x_j) = s_{km}$ za vse $s_i \in S_2$

- Operator za določanje izhodov:
Za $s_{km} \in S_1$ sledi $\mu'(s_{km}) = y_m$, za vsako $s_i \in S_2$ pa sledi $\mu'(s_i) = -$ (izhod ni določen)

Primer 52: Pretvorba sekvenčnega stroja tipa Mealy v sekvenčni stroj tipa Moore

- Naredimo kanonsko tabelo za sekvenčni stroj tipa Moore z enakim številom stolpcev za stanja, v katera sekvenčni stroj prehaja, kot jih ima sekvenčni stroj tipa Mealy in z dodatnim stolpcem za izhodne črke.

- Za vsak par stanje/izhodna črka v kanonski tabeli sekvenčnega stroja tipa Mealy zapišemo novo stanje v prvi stolpec kanonske tabele za sekvenčni stroj tipa Moore. Oznake novih stanj vsebujejo tako oznako stanja sekvenčnega stroja tipa Mealy kot tudi oznako izhodne črke, na primer: $1/2 \Rightarrow s_{12}$.

- Če sekvenčni stroj tipa Mealy v kakšno od stanj ne prehaja (stanje je zapisano samo v levem stolpcu), stanje vseeno dodamo med stanja sekvenčnega stroja tipa Moore.

- Sledi določanje prehodov med stanji sekvenčnega stroja tipa Moore: za stanje s_{AI} pogledamo kam iz stanja s_A pri izbrani vhodni črki prehaja sekvenčni stroj tipa Mealy. Če prehaja v B/J, v kanonsko tabelo sekvenčnega stroja tipa Moore zapišemo BJ.

- Izhodne črke sekvenčnega stroja tipa Moore določa drugi del oznake stanja. Za stanja z enim indeksom (začetna stanja) izhodna črka ni določena.

- Po potrebi uvedemo nove oznake stanj.

Mealy:

	x_1	x_2
s_1	2/2	1/2
s_2	3/2	1/2
s_3	4/1	1/2
s_4	4/2	1/3

$\Rightarrow$

Moore:

	x_1	x_2	
s_{12}	22	12	2
s_{13}	22	12	3
s_{22}	32	12	2
s_{32}	41	12	2
s_{41}	42	13	1
s_{42}	42	13	2

$\Rightarrow$

Moore:

	x_1	x_2	
s_1	3	1	2
s_2	3	1	3
s_3	4	1	2
s_4	5	1	2
s_5	6	2	1
s_6	6	2	2

Primer 53: Pretvorba sekvenčnega stroja tipa Mealy z redundancami v sekvenčni stroj tipa Moore

Mealy:

	x_1	x_2
s_1	2/2	4/1
s_2	3/–	–
s_3	2/1	1/–
s_4	–	–

$\Rightarrow$

Moore:

	x_1	x_2	
s_{1-}	22	41	–
s_{21}	3–	–	1
s_{22}	3–	–	2
s_{3-}	21	1–	–
s_{41}	–	–	1

$\Rightarrow$

Moore:

	x_1	x_2	
s_1	3	5	–
s_2	4	–	1
s_3	4	–	2
s_4	2	1	–
s_5	–	–	1

12.3.2 Pretvorba Moore ⇒ Mealy

Zapišimo sekvenčni stroj tipa Moore (M) s petorčkom $< X, Y, S, \lambda, \mu >$ in sekvenčni stroj tipa Mealy (M') s petorčkom $< X', Y', S', \lambda', \delta' >$. Potem velja:

- Vhodne črke, izhodne črke in stanja:

$$\begin{aligned} X' &= X \\ Y' &= Y \\ S' &= S \end{aligned}$$

- Logični funkciji za prehajanje stanj in določanje izhodov:
  Če je $\lambda(s_i, x_j) = s_k$ in $\mu(s_k) = y_m$, potem je $\lambda'(s_i, x_j) = s_k$ in $\delta'(s_i, x_j) = y_m$.

Primer 54: Pretvorba sekvenčnega stroja tipa Moore v sekvenčni stroj tipa Mealy

- Naredimo kanonsko tabelo za sekvenčni stroj tipa Mealy z enakim številom vrstic (stanj) in stolpcev s stanji, v katera sekvenčni stroj prehaja, vendar brez stolpca, ki podaja izhodne črke.
- V kanonsko tabelo za sekvenčni stroj tipa Mealy prepišemo stanja iz kanonske tabele za sekvenčni stroj tipa Moore.
- Stanjem v kanonski tabeli za sekvenčni stroj tipa Mealy dodamo ustrezne izhodne črke. Za vsako stanje, v katerega sekvenčni stroj prehaja, jih odčitamo v kanonski tabeli za sekvenčni stroj tipa Moore.

Moore:

	x_1	x_2	
s_1	1	2	y_1
s_2	4	3	y_2
s_3	4	3	y_2
s_4	2	1	y_1

$\Rightarrow$

Mealy:

	x_1	x_2
s_1	1/1	2/2
s_2	4/1	3/2
s_3	4/1	3/2
s_4	2/2	1/1

Primer 55: Pretvorba sekvenčnega stroja tipa Moore z redundancami v sekvenčni stroj tipa Mealy

Moore:

	x_1	x_2	
s_1	2	1	y_1
s_2	3	2	y_2
s_3	4	–	–
s_4	1	–	y_3

$\Rightarrow$

Mealy:

	x_1	x_2
s_1	2/2	1/1
s_2	3/–	2/2
s_3	4/3	–
s_4	1/1	–

Poglavje 13

Minimizacija stanj sekvenčnih strojev

Realizacija sekvenčnih strojev je enostavnejša in cenejša, če ima sekvenčni stroj manj stanj. Zato je pred realizacijo število stanj smiselno zmanjšati na minimum.

Ideja minimizacije je v združevanju stanj z ekvivalentnim obnašanjem. Dve stanji lahko združimo če imata enaki izhodni črki in prehajata v isto skupino stanj. Ogledali si bomo minimizacijo stanj sekvenčnih strojev z metodo kompatibilnih razredov in z metodo trikotne tabele.

13.1 Metoda kompatibilnih razredov

- V kanonski tabeli sekvenčnega stroja opazujemo pare stanj. Če na vse vhodne črke odgovarjajo kompatibilno (Mealy) oziroma, če imajo kompatibilne izhodne črke (Moore) in če prehajajo v kompatibilne pare stanj ali posamezna stanja, potem so to pravi kompatibilni pari. Sicer jih izločimo iz nadaljnje obravnave. Kompatibilnost pomeni enakost v primeru obeh definiranih izhodov oziroma v primeru, ko je vsaj eden od obeh izhodov redundanten.

- Prave kompatibilne pare poskušamo združevati v razrede z upoštevanjem naslednjih lastnosti:

  - Minimalnost: Pomeni, da postopek ponavljamo dokler je mogoče.

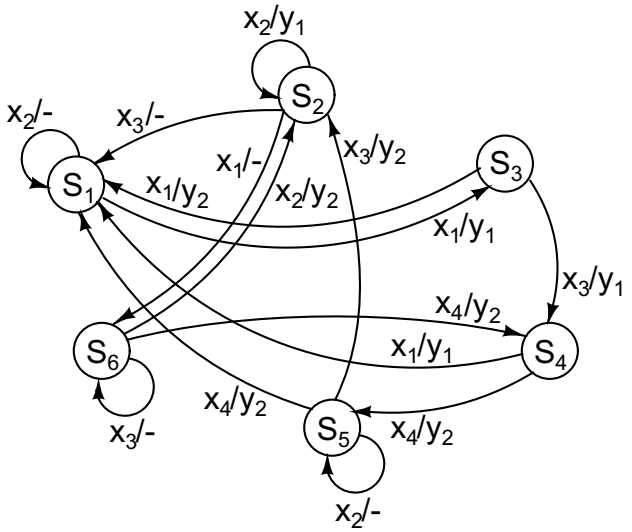
  - Polnost: Pomeni, da morajo biti vsa stanja zajeta vsaj v enem razredu, če ne gre drugače lahko tvorijo svoj razred.

  - Zaprtost: Je najbolj restriktivna lastnost, saj zahteva, da razredi prehajajo v razrede enoumno. Slednje pomeni, da sekvenčni stroj lahko prehaja v času $t+1$ le v enega od razredov, ki nastopa v času t ali kvečjemu v podmnožico razreda (enega ali več). Nikakor pa ne sme biti razred v katerega sekvenčni stroj prehaja v času $t+1$ množica, ki delno pripada enemu, delno pa drugemu razredu iz časa t.

Primer 56: Minimizacija stanj sekvenčnega stroja tipa Mealy z redundancami

	x_1	x_2	x_3	x_4
s_1	$3/1$	$1/-$	$-$	$-$
s_2	$6/-$	$2/1$	$1/-$	$-$
s_3	$1/2$	$-$	$4/1$	$-$
s_4	$1/1$	$-$	$-$	$5/2$
s_5	$-$	$5/-$	$2/2$	$1/2$
s_6	$-$	$2/2$	$6/-$	$4/2$

Sekvenčni stroj podan s kanonsko tabelo lahko predstavimo tudi z diagramom prehajanja stanj



Najprej poiščemo vse pare stanj, ki imajo pri vseh vhodnih črkah kompatibilne izhodne črke. Dve izhodni črki sta kompatibilni, če sta enaki oziroma če vsaj ena ni podana. Na primer, stanji s_1 in s_2 imata kompatibilne izhodne črke: izhodni črki y_1 in $-$ pri vhodni črki x_1 sta kompatibilni, enako velja pri vhodni črki x_2. Pri vhodnih črkah x_3 in x_4 pa izhodne črke niso definirane. Ta par stanj zato vpišemo v levi stolpec spodnje tabele. V desne stolpce v tem primeru za vsako vhodno črko posebej vpišemo kombinacije stanj v katera par stanj prehaja: stanje s_1 prehaja pri vhodni črki x_1 v s_3, stanje s_2 pa v s_6, zato na ustrezno mesto v spodnji tabeli vpišemo $3,6$. Po drugi strani pa stanji s_1 in s_3 nimata kompatibilnih izhodnih črk, saj je pri vhodni črki x_1 izhodna črka enkrat y_1, drugič pa y_2. Tega para stanj zato ne vpišemo v levi stolpec spodnje tabele. Ko na ta način pregledamo vse pare stanj, dobimo naslednjo tabelo

	x_1	x_2	x_3	x_4
$1,2$	$3,6$	$1,2$	1	$-$
$1,4$	$1,3$	1	$-$	5
$1,5$	3	$1,5$	2	1
$1,6$	3	$1,2$	6	4
$2,3$	$1,6$	2	$1,4$	$-$
$2,4$	$1,6$	2	1	5
$2,5$	6	$2,5$	$1,2$	1
$3,6$	1	2	$4,6$	4
$4,5$	1	5	2	$1,5$
$4,6$	1	2	6	$4,5$
$5,6$	$-$	$2,5$	$2,6$	$1,4$

Nato v vsaki vrstici pogledamo, če so pari stanj v katera sekvenčni stroj prehaja navedeni v levem stolpcu. Če stroj prehaja samo v eno stanje, se mora to stanje pojaviti vsaj v enem od parov stanj v levem stolpcu. Primer: par stanj $1,4$ prehaja pri vhodni črki x_1 v par stanj $1,3$. Ker ta par ne obstaja v levem stolpcu, vrstico enostavno brišemo iz tabele. Po drugi strani pa par stanj $1,5$ prehaja pri vhodni črki x_1 v stanje 3. To stanje je navedeno v paru $3,6$. V tabeli najdemo pare tudi za vse ostale vhodne črke, zato je par stanj $1,5$ pravi kompatibilni par in ostane v tabeli. Postopek ponavljamo, dokler nam v tabeli ne ostanejo samo pravi kompatibilni pari.

		x_1	x_2	x_3	x_4
	$1,2$	$3,6$	$1,2$	1	$-$
$\times$	$1,4$	$1,3$	1	$-$	5
	$1,5$	3	$1,5$	2	1
	$1,6$	3	$1,2$	6	4
$\times$	$2,3$	$1,6$	2	$1,4$	$-$
	$2,4$	$1,6$	2	1	5
	$2,5$	6	$2,5$	$1,2$	1
	$3,6$	1	2	$4,6$	4
	$4,5$	1	5	2	$1,5$
	$4,6$	1	2	6	$4,5$
$\times$	$5,6$	$-$	$2,5$	$2,6$	$1,4$

$\Rightarrow$

	x_1	x_2	x_3	x_4
$1,2$	$3,6$	$1,2$	1	$-$
$1,5$	3	$1,5$	2	1
$1,6$	3	$1,2$	6	4
$2,4$	$1,6$	2	1	5
$2,5$	6	$2,5$	$1,2$	1
$3,6$	1	2	$4,6$	4
$4,5$	1	5	2	$1,5$
$4,6$	1	2	6	$4,5$

Pare stanj poskušamo združiti v čim večje razrede, Minimizacija je tem bolj uspešna čim manj je razredov. Primer: para $1,2$ in $1,5$ prehajata pri x_1 v stanje $3,6$ ($3,6$ je edini par, ki vključuje stanje 3), pri x_2 oba prehajata vase, ... Enako lahko ugotovimo tudi za para $1,2$ in $2,5$. Zato lahko stanja $1,2,5$ združimo v en razred. Označili ga bomo z A. Spodnja tabela prikazuje rezultat združevanja

	x_1	x_2	x_3	x_4
$A=1,2,5$	$B/1$	$A/1$	$A/2$	$A/2$
$B=3,6$	$A/2$	$A/2$	$C/1$	$(C \vee D)/2$
$C=4,6$	$A/1$	$A/2$	$(B \vee C)/-$	$D/2$
$D=4,5$	$A/1$	$(A \vee D)/-$	$A/2$	$A/2$

Na prvi pogled se zdi, da razred D ni potreben, saj sta obe stanji $4,5$ že v razredih A in C. Vendar tedaj kombinacija (C, x_4) ne bi vodila v določeno stanje, saj pogoj zaprtosti ne bi bil izpolnjen.
Nazadnje še zamenjano oznake razredov z običajnimi oznakami:

	x_1	x_2	x_3	x_4
s_1	$2/1$	$1/1$	$1/2$	$1/2$
s_2	$1/2$	$1/2$	$3/1$	$(3 \vee 4)/2$
s_3	$1/1$	$1/2$	$(2 \vee 3)/-$	$4/2$
s_4	$1/1$	$(1 \vee 4)/-$	$1/2$	$1/2$

Pravilnost postopka minimizacije sekvenčnega stroja lahko preverimo s testnim zaporedjem. To mora biti zaradi redundanc prilagojeno osnovnemu sekvenčnemu stroju.
Poglejmo odgovore originalnega in minimiziranega sekvenčnega stroja pri zaporedju vhodnih črk: $x_2, x_1, x_3, x_4, x_4, x_2, x_1, x_1$:

- Originalni sekvenčni stroj:

$$\begin{array}{lccccccccccccccccc} x(t): & & x_2 & & x_1 & & x_3 & & x_4 & & x_4 & & x_2 & & x_1 & & x_1 & \\ s(t): & S_1 & \Longrightarrow & S_1 & \Longrightarrow & S_3 & \Longrightarrow & S_4 & \Longrightarrow & S_5 & \Longrightarrow & S_1 & \Longrightarrow & S_1 & \Longrightarrow & S_3 & \Longrightarrow & S_1 \\ y(t): & & & - & & y_1 & & y_1 & & y_2 & & y_2 & & - & & y_1 & & y_2 \end{array}$$

- Minimizirani sekvenčni stroj:

$$\begin{array}{lccccccccccccccccc} x(t): & & x_2 & & x_1 & & x_3 & & x_4 & & x_4 & & x_2 & & x_1 & & x_1 & \\ s(t): & S_1 & \Longrightarrow & S_1 & \Longrightarrow & S_2 & \Longrightarrow & S_3 & \Longrightarrow & S_4 & \Longrightarrow & S_1 & \Longrightarrow & S_1 & \Longrightarrow & S_2 & \Longrightarrow & S_1 \\ y(t): & & & y_1 & & y_1 & & y_1 & & y_2 & & y_2 & & y_1 & & y_1 & & y_2 \end{array}$$

Pari izhodnih črk originalnega in minimiziranega sekvenčnega stroja so kompatibilni, iz tega lahko sklepamo, da sta sekvenčna stroja ekvivalentna.

13.2 Metoda trikotne tabele

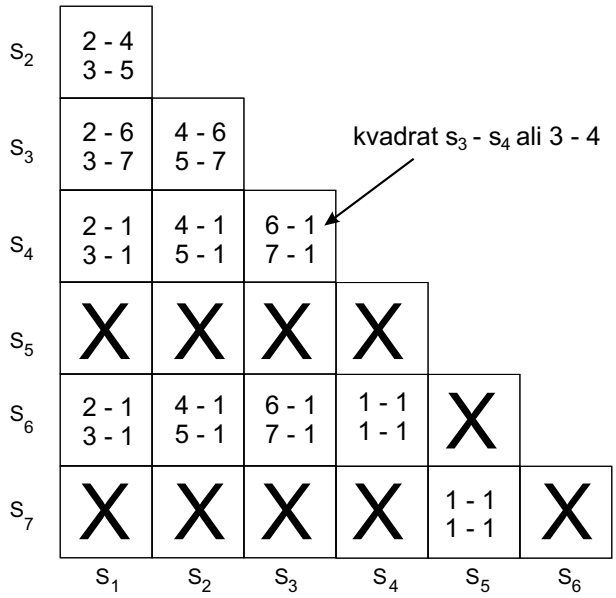
Postopek minimizacije:

- Za sekvenčni stroj, ki ima K stanj, formiramo trikotno tabelo s $K-1$ stolpci in $K-1$ vrsticami. Stolpce označujemo spodaj od leve proti desni $(s_1, s_2, \ldots, s_{K-1})$, vrstice pa levo od zgoraj navzdol $(s_2, s_3, \ldots, s_K)$. Vsak kvadrat v tabeli predstavlja par stanj.
- Če par stanj nima kompatibilnih izhodnih črk, ustrezen kvadrat prečrtamo (vpišemo $\times$). Drugače v kvadrat za obe stanji vpišemo v kateri stanji sekvenčni stroj prehaja – pare stanj, v katera sekvenčni stroj prehaja, pišemo za vsako vhodno črko v svojo vrstico.
- Prečrtamo tudi kvadrate s tistimi pari stanj, ki imajo v trikotni tabeli že prečrtan par (par z oznako $\times$).
- Na koncu parom, ki vodijo v neprečrtane pare stanj in parom v tabeli dodelimo nove oznake in zapišemo kanonsko tabelo minimiziranega sekvenčnega stroja.

Primer 57: Minimizirajmo sekvenčni stroj, ki zazna, kdaj se pojavi eno od zaporedij vhodnih črk $x_1x_2x_1$ ali $x_2x_2x_1$.
Ustrezen sekvenčni stroj tipa Mealy podaja kanonska tabela

	x_1	x_2
s_1	2/1	3/1
s_2	4/1	5/1
s_3	6/1	7/1
s_4	1/1	1/1
s_5	1/2	1/1
s_6	1/1	1/1
s_7	1/2	1/1

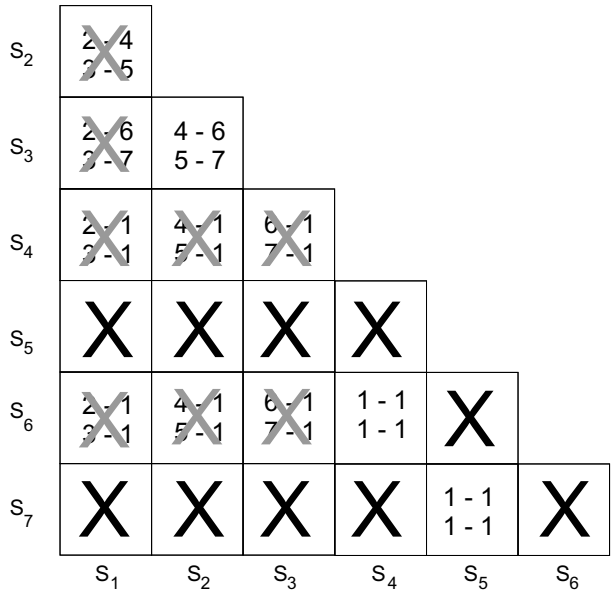
Pripravimo ustrezno trikotno tabelo:



Stanji s_1 in s_5 se razlikujeta v izhodnih črkah, zato kvadrat $1-5$ prečrtamo. Po drugi strani pa imata, na primer, stanji s_2 in s_3 pri vseh vhodnih črkah enake izhodne črke, zato v kvadrat $2-3$ vpišemo pare stanj, v katera prehajata. Ker pri vhodni črki x_1 stanje s_2 preide v stanje s_4, stanje s_3 pa v stanje s_6 v kvadrat vpišemo par $4-6$. Podobno za vhodno črko x_2 dobimo par $5-7$. Na enak način zapolnimo celo tabelo.

Nato za vsak par $i-j$ v neprečrtanih kvadratih pogledamo, če je kvadrat $i-j$ prečrtan. Če je, prečrtamo tudi obravnavani kvadrat. Kvadrat $1-2$, na primer, moramo prečrtati, saj je kvadrat $3-5$ že prečrtan. Pozor! Pari enakih števil $i-i$ v tabeli pomenijo, da obe stanji, ki jih opisuje ustrezen kvadrat pri določeni vhodni črki prehajata v isto stanje. Na podlagi takih parov obravnavanega kvadrata nikoli ne črtamo.

Postopek črtanja ponavljamo dokler ne moremo prečrtati nobenega kvadrata več. Na koncu dobimo naslednjo trikotno tabelo:



Stanja, ki določajo neprečrtane kvadrate so ekvivalentna. Ekvivalentnim stanjem dodelimo nove oznake

$$\begin{aligned} s_2 &\equiv s_3 \equiv s_2' \quad , \\ s_4 &\equiv s_6 \equiv s_3' \quad , \\ s_5 &\equiv s_7 \equiv s_5' \quad \text{in} \end{aligned}$$

zapišemo kanonsko tabelo minimiziranega sekvenčnega stroja. Dobimo jo tako, da v kanonski tabeli osnovnega sekvenčnega stroja zamenjamo oznake, vrstice, ki se ponovijo pa izpustimo. Z dodatnim preimenovanjem stanj $s_2' \Rightarrow s_2$, $s_4' \Rightarrow s_3$ in $s_5' \Rightarrow s_4$ dobimo običajno kanonsko tabelo za minimizirani sekvenčni stroj.

	x_1	x_2
s_1	$2'/1$	$2'/1$
s_2'	$4'/1$	$5'/1$
s_4'	$1/1$	$1/1$
s_5'	$1/2$	$1/1$

$\Rightarrow$

	x_1	x_2
s_1	$2/1$	$2/1$
s_2	$3/1$	$4/1$
s_3	$1/1$	$1/1$
s_4	$1/2$	$1/1$

Poglavje 14

Strukturna sinteza sekvenčnih strojev

Strukturna sinteza je postopek, ki nas privede od kanonske tabele ali diagrama prehajanja stanj sekvenčnega stroja do logične sheme. Slednja predstavlja logično realizacijo sekvenčnega stroja.
Postopek:

- Kodiranje vhodne, izhodne in notranje abecede.
- Izdelava aplikacijske tabele.
- Izbira pomnilnih celic.
- Določitev krmilnih in izhodnih funkcij.
- Po potrebi minimizacija krmilnih in izhodnih funkcij.
- Realizacija odločitvene logike z izbranimi operatorji.

Primer 58: Oglejmo si strukturno sintezo sekvenčnega stroja, ki smo ga minimizirali v poglavju 13.1. Kanonska tabela minimiziranega sekvenčnega stroja:

	x_1	x_2	x_3	x_4
s_1	$2/1$	$1/1$	$1/2$	$1/2$
s_2	$1/2$	$1/2$	$3/1$	$(3 \vee 4)/2$
s_3	$1/1$	$1/2$	$(2 \vee 3)/-$	$4/2$
s_4	$1/1$	$(1 \vee 4)/-$	$1/2$	$1/2$

- Določimo najmanjše potrebno število vhodnih, izhodnih in notranjih spremenljivk. Iz relacije med številom črk v abecedi in potrebnim številom spremenljivk iz poglavja 12.1 sledi
  - število vhodnih spremenljivk (x_1, x_2, x_3, x_4): $N = 4 \Rightarrow n = \lceil \log_2 4 \rceil = 2$
  - število izhodnih spremenljivk (y_1, y_2): $M = 2 \Rightarrow m = \lceil \log_2 2 \rceil = 1$
  - število notranjih spremenljivk – stanj (s_1, s_2, s_3, s_4): $K = 4 \Rightarrow k = \lceil \log_2 4 \rceil = 2$

  Vhodno in izhodno abecedo bomo kodirali poljubno (po vrsti),

	ξ_1	ξ_2
x_1	0	0
x_2	0	1
x_3	1	0
x_4	1	1

	η_1
y_1	0
y_2	1

pri notranji abecedi pa bomo upoštevali vsebino kanonske tabele. Pari stanj (s_1, s_4), (s_3, s_4) in (s_2, s_3) v kanonski tabeli nastopajo v disjunkcijah, zato je smiselno, da parom določimo kode, ki se razlikujejo samo v eni spremenljivki

	γ_1	γ_2
s_1	0	0
s_2	0	1
s_3	1	1
s_4	1	0

- Izdelamo aplikacijsko tabelo. V aplikacijski tabeli na levi strani nastopajo vse vhodne in notranje spremenljivke, na sredini nove notranje spremenljivke in na desni izhodne spremenljivke. Na levi strani napišemo vse možne kombinacije vhodnih in notranjih spremenljivk, stolpce na sredini in na desni pa izpolnimo z uporabo kanonske tabele.

ξ_1	ξ_2	$\gamma_1(t)$	$\gamma_2(t)$	$\gamma_1(t+1)$	$\gamma_2(t+1)$	η_1
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	×	0	×
0	1	1	1	0	0	1
1	0	0	0	0	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	×	1	×
1	1	0	0	0	0	1
1	1	0	1	1	×	1
1	1	1	0	0	0	1
1	1	1	1	1	0	1

Podrobno si poglejmo prvo vrstico aplikacijske tabele. Iz kodirnih tabel je razvidno, da vhodni spremenljivki $\xi_1\xi_2 = 00$ določata vhodno črko x_1, notranji spremenljivki $\gamma_1(t)\gamma_2(t) = 00$ pa stanje s_1. V kanonski tabeli lahko preberemo, da gre sekvenčni stroj iz trenutnega stanja s_1 z vhodno črko x_1 v novo stanje s_2, pri čemer se na izhodu pojavi izhodna črka y_1. Iz kodirnih tabel naprej preberemo, da je stanj s_2 zapisano s spremenljivkama $\gamma_1(t+1)\gamma_2(t+1) = 01$, izhodna črka y_1 pa s spremenljivko $\eta_1 = 0$. Vrednosti spremenljivk vpišemo v aplikacijsko tabelo in nadaljujemo z naslednjo vrstico. Zaradi pametnega izbora kod za notranja stanja dobimo v nekaterih vrsticah redundance (×).

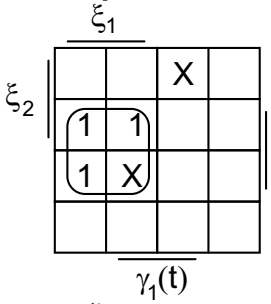
- Sekvenčni stroj bomo realizirali s pomnilnimi celicami D. Ker imamo dve notranji spremenljivki potrebujemo dve pomnilni celici D.

- K aplikacijski tabeli dopišemo dva stolpca z vhodi v pomnilni celici D, D_1 in D_2. Za pomnilno celico D velja, da moramo dati na vhod tisto vrednost, ki naj si jo zapomni, v našem primeru je tako $D_i = \gamma_i(t+1)$, $i = 1, 2$. Pri bolj kompleksnih pomnilnih celicah si lahko pomagamo tudi s tabelo prehajanja stanj, ki smo jo zgradili v poglavju 11.3.

ξ_1	ξ_2	$\gamma_1(t)$	$\gamma_2(t)$	$\gamma_1(t+1)$	$\gamma_2(t+1)$	η_1	D_1	D_2
0	0	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0
0	1	1	0	×	0	×	×	0
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	1	0	0
1	0	0	1	1	1	0	1	1
1	0	1	0	0	0	1	0	0
1	0	1	1	×	1	×	×	1
1	1	0	0	0	0	1	0	0
1	1	0	1	1	×	1	1	×
1	1	1	0	0	0	1	0	0
1	1	1	1	1	0	1	1	0

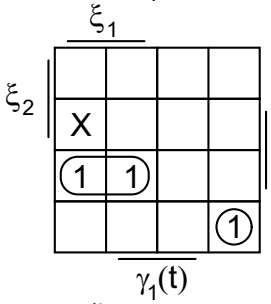
- Krmilne in izhodne funkcije še minimiziramo

$D_1 = \gamma_1(t+1)$:



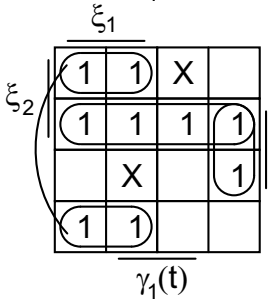
$\gamma_1(t+1) = \xi_1\gamma_2(t)$

$D_2 = \gamma_2(t+1)$:



$\gamma_2(t+1) = \xi_1\overline{\xi}_2\gamma_2(t) \vee \overline{\xi}_1\overline{\xi}_2\overline{\gamma}_1(t)\overline{\gamma}_2(t)$

η_1:



$\eta_1 = \xi_1\overline{\gamma}_2(t) \vee \xi_2\overline{\gamma}_2(t) \vee \overline{\xi}_1\overline{\gamma}_1(t)\gamma_2(t)$

- in narišemo logično shemo vezja