4. PREKLOPNI MNOGOPOLI

4.1 Matrično opisovanje vezij

4.1.1 Osnovna simbolika

Vektor:

$\mathbf{a}_n$ – vodoravni vektor

$\mathbf{a}^m$ – navpični vektor

Matrika:

$\mathbf{A}_{1:n}^{1:m}$ - matrika reda m x n

$\mathbf{A}_{1:m}^{1:n}$ - matrika reda n x m

$\mathbf{C}_{1:(m+n)}^{1:t} = \mathbf{A}_{1:m}^{1:t}\ \mathbf{B}_{1:n}^{1:t}$ - vodoravni niz matrik **A** in **B**

$\mathbf{C}_{1:t}^{1:(m+n)} = \begin{matrix}\mathbf{A}_{1:t}^{1:m} \\ \mathbf{B}_{1:t}^{1:n}\end{matrix}$ - navpični niz matrik **A** in **B**

$\mathbf{C}_{1:m}^{1:n} = \left[\mathbf{A}_{1:n}^{1:m}\right]^T;\ c_j^i = a_i^j$ - transpozicija matrike **C**

4.1.2 Operacije nad vektorji in matrikami

Negacija:

$$\mathbf{C}_n^m = \overline{\mathbf{A}}_n^m; \quad c_j^i = \overline{a}_j^i$$

Konjunkcija:

$$\mathbf{C}_n^m = \mathbf{A}_n^m \bullet \mathbf{B}_n^m; \quad c_j^i = a_j^i \bullet b_j^i$$

Disjunkcija:

$$\mathbf{C}_n^m = \mathbf{A}_n^m + \mathbf{B}_n^m; \quad c_j^i = a_j^i + b_j^i$$

Izključna ALI operacija:

$$\mathbf{C}_n^m = \mathbf{A}_n^m \oplus \mathbf{B}_n^m; \quad c_j^i = a_j^i \oplus b_j^i$$

$$c = \circ \mid \mathbf{a}_n; \qquad c = \circ \mid \mathbf{a}^m,$$

kjer je:

° - splošni znak za operator,

$c = a_1 \circ a_2 \circ a_3 \circ a_4 \circ \ldots\ldots \circ a_n$ – vodoravna redukcija vektorja k skalarju,

$c = a^1 \circ a^2 \circ a^3 \circ a^4 \circ \ldots\ldots \circ a^m$ – navpična redukcija vektorja k skalarju.

$$\mathbf{C}_{1:m}^{1:n} = \mathbf{A}_{1:t}^{1:n} *\circ \mathbf{B}_{1:m}^{1:t}$$

$$c_j^i = *\left| \left(a_{1:t}^i \circ b_i^{1:t} \right) \right.$$

Elemente vrstice v matriki **A** povežemo z operacijo »°« z elementi stolpca matrike **B**. Nato pa z vektorsko redukcijo preidemo preko operacije » *« na matriko **C**.

4. 2 Popolna disjunktivna normalna oblika vektorske preklopne funkcije

$$f(x_1\ x_2, \ldots\ldots\ldots, x_n) = \sum_{i=0}^{2^n-1} f_i\, m_i\ ;$$

$$\mathbf{x} = [x_1, x_2, \ldots\ldots, x_n],$$

$$\mathbf{m} = \left[m_0, m_1, \ldots\ldots m_{2^n-1} \right]$$

$$\mathbf{f} = \left[f_0, f_1, \ldots\ldots\ldots f_{(2^n-1)} \right]^T$$

$$y = f(\mathbf{x}) = \mathbf{m}\ \Sigma\&\ \mathbf{f}$$

Mintermski vektor **m** je dan z operacijo konjunkcije in ekvivalence:

$$\mathbf{m}(\mathbf{x}) = \mathbf{x}\ \&\ \equiv \mathbf{W}^T,$$

$$y = f(\mathbf{x}) = [\ \mathbf{x}\ \&\ \equiv \mathbf{W}^T]\ \sum\ \&\ \mathbf{f}$$

x	y = f (**x**)
W	**f**

f(**x**) preide v vektorsko funkcijo **y,** z njo pa preide tudi vektor **f** v matriko **D**.

Torej:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = [\ \mathbf{x}\ \&\ \equiv \mathbf{W}^T]\ \sum\ \&\ \mathbf{D}$$ – matrična popolna disjunktivna normalna

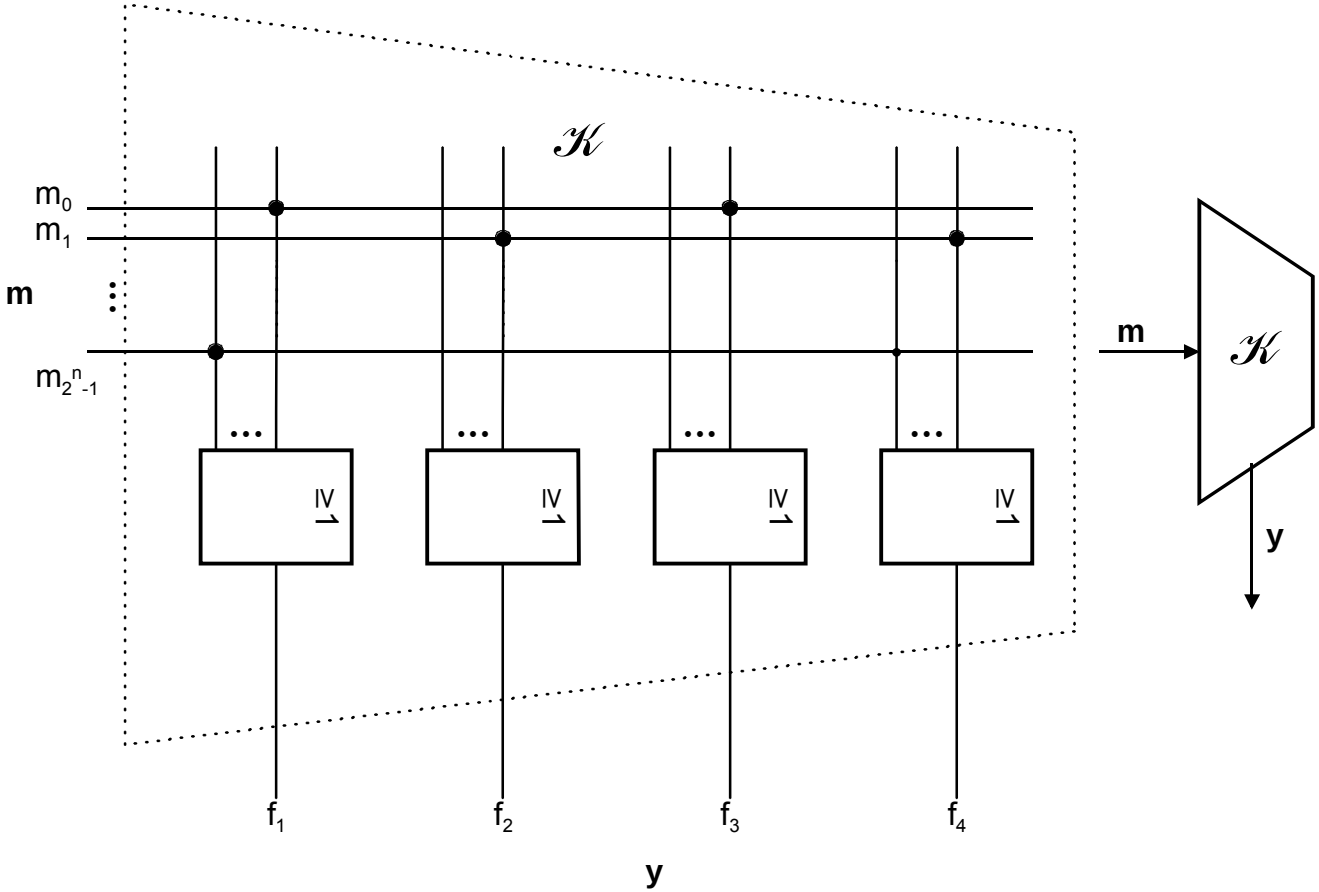
Pri vektorski preklopni funkciji dobi tudi pravilnostna tabela nove simbole:

x	**y = f (x)**
W	**D**

4. 3 Standardna mnogopolna preklopna vezja

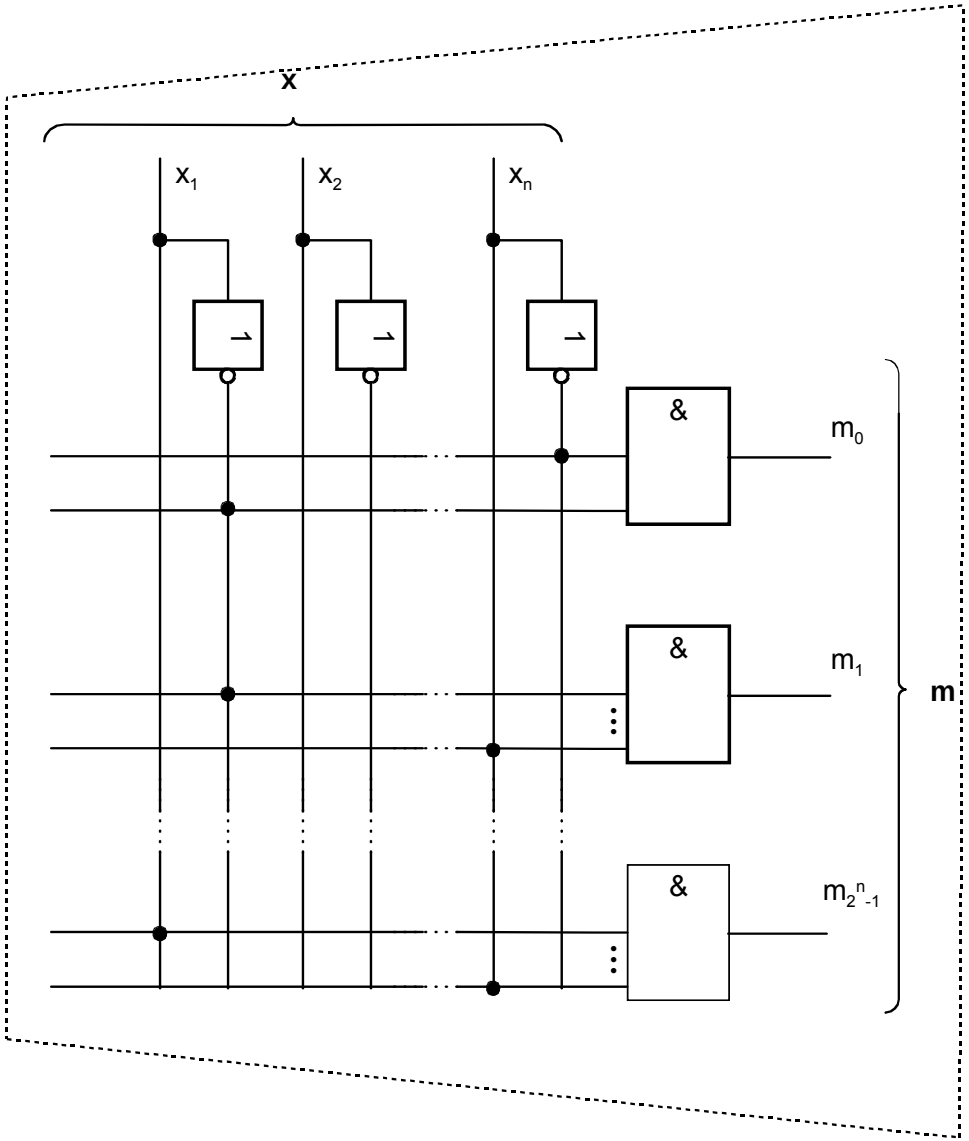
4.3.1 Kodirna vezja – kodirniki (encoder-ji)

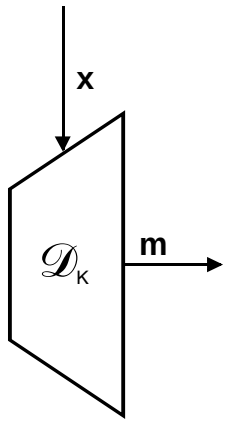
$\mathbf{y = m}\,\Sigma\,\&\,\mathbf{K}$; k_j^i = konstanta



V konstantah matrike **K** je zapisano kodirno pravilo.

4. 3. 2 Dekodirna vezja – dekodirniki (decoder-ji)

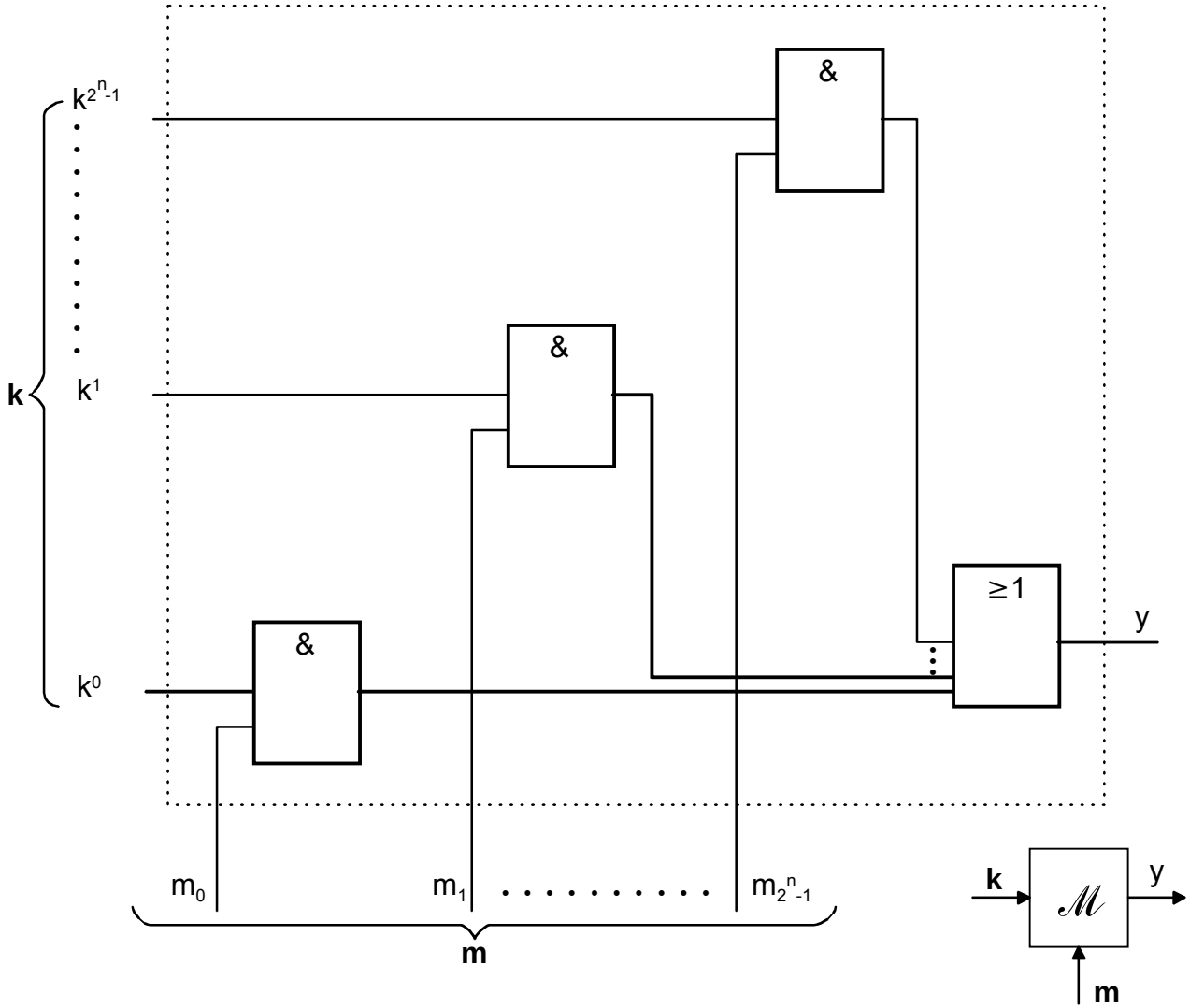




$\mathbf{m} = \mathbf{x}\ \& \equiv \mathbf{W}^{\mathbf{T}}$; w_j^i = konstanta $\qquad$ $\mathbf{m} = \mathbf{x}\ \& \equiv \mathbf{D}^{\mathbf{T}}$; d_j^i = konstanta

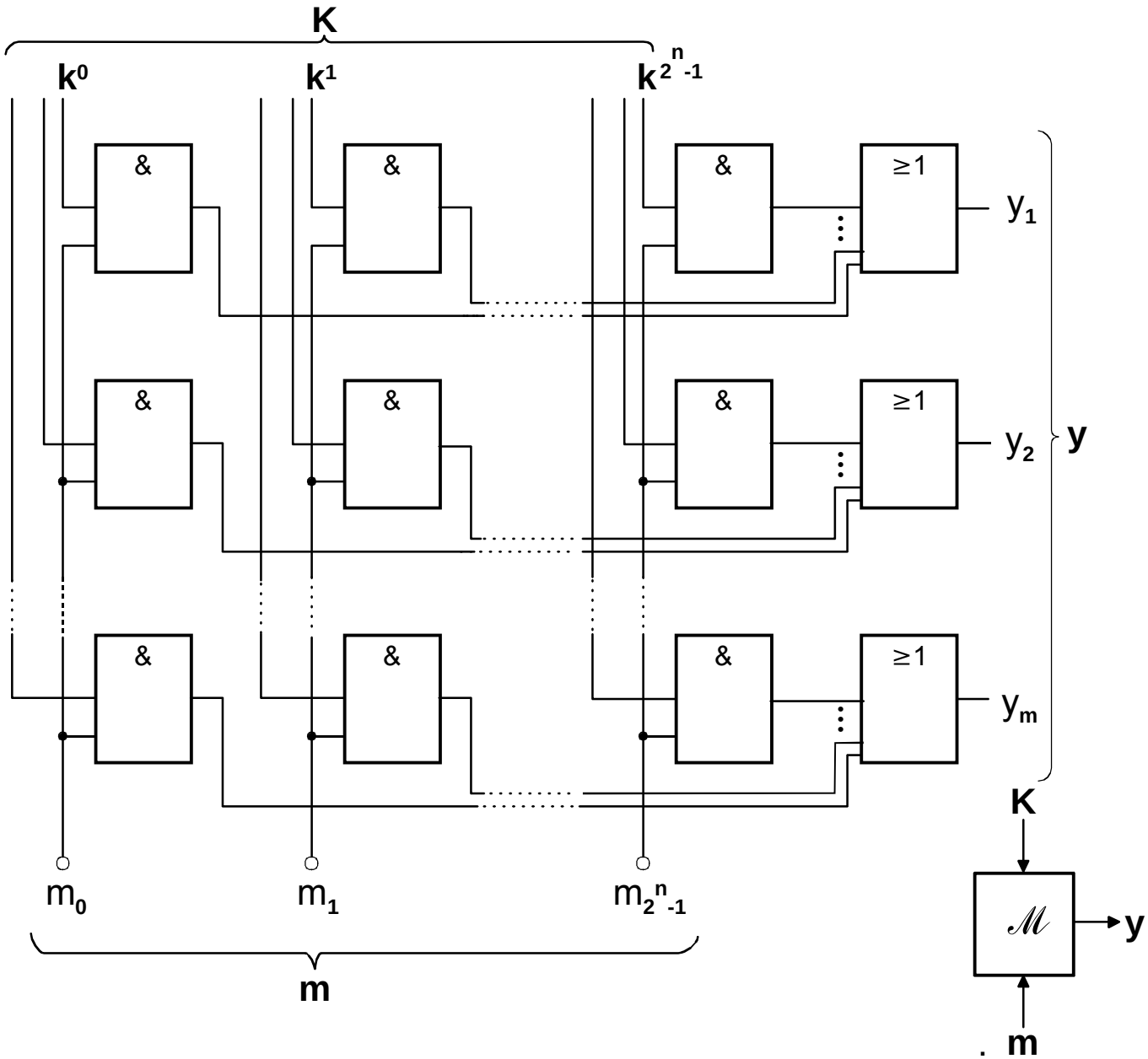
4. 3. 3 Multipleksor (Multiplexer)

4. 3. 3.1 Skalarni neadresni multipleksor



$y = \mathbf{m}\,\Sigma\,\&\,\mathbf{k}$; k^i = spremenljivka

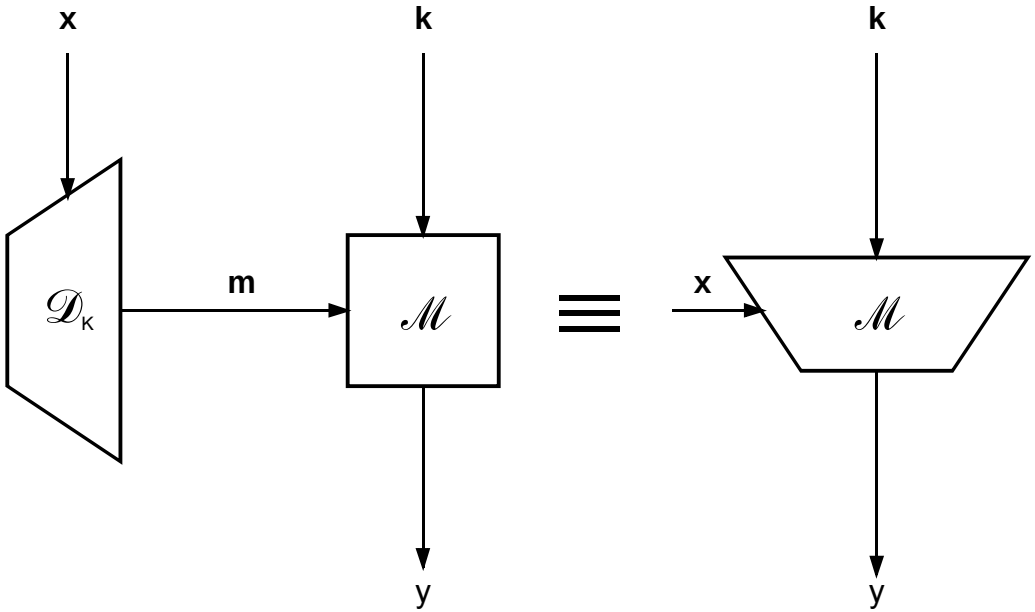
4. 3. 3. 2 Vektorski neadresni multipleksor



$\mathbf{y} = \mathbf{m}\,\Sigma\,\&\,\mathbf{K}$; $k^i_j =$ spremenljivke, ki tvorijo vektorje k^i

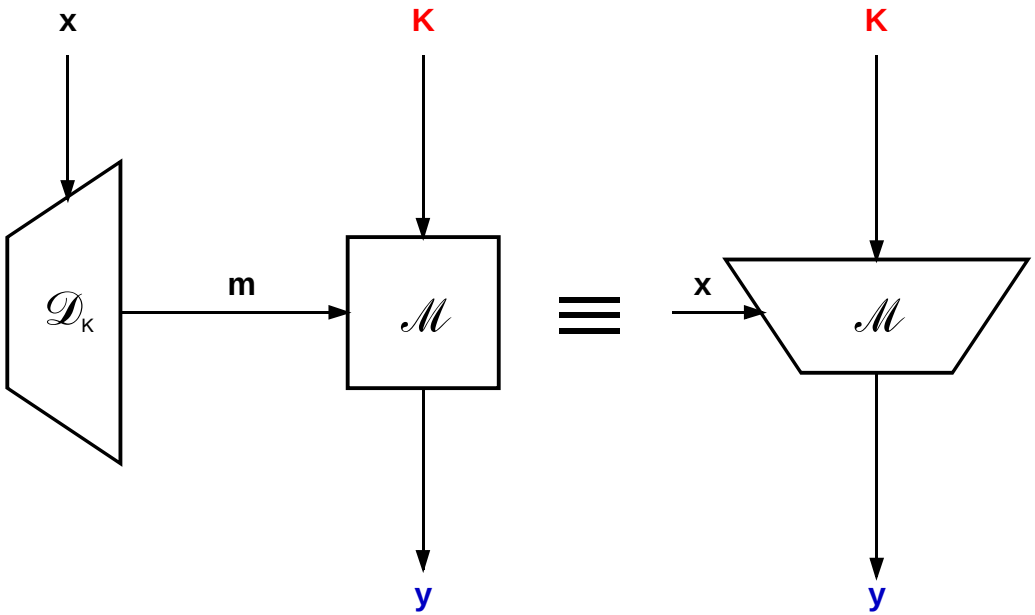
4. 3. 3. 3 Skalarni adresni multipleksor

Dve elementarni operaciji: dekodiranje - multipleksiranje



$y = (\mathbf{x} \,\&\, \equiv \mathbf{W}^{\mathbf{T}})\Sigma \,\&\, \mathbf{k}$; k^i = spremenljivka - skalarna

4. 3. 3. 4 Vektorski adresni multipleksor

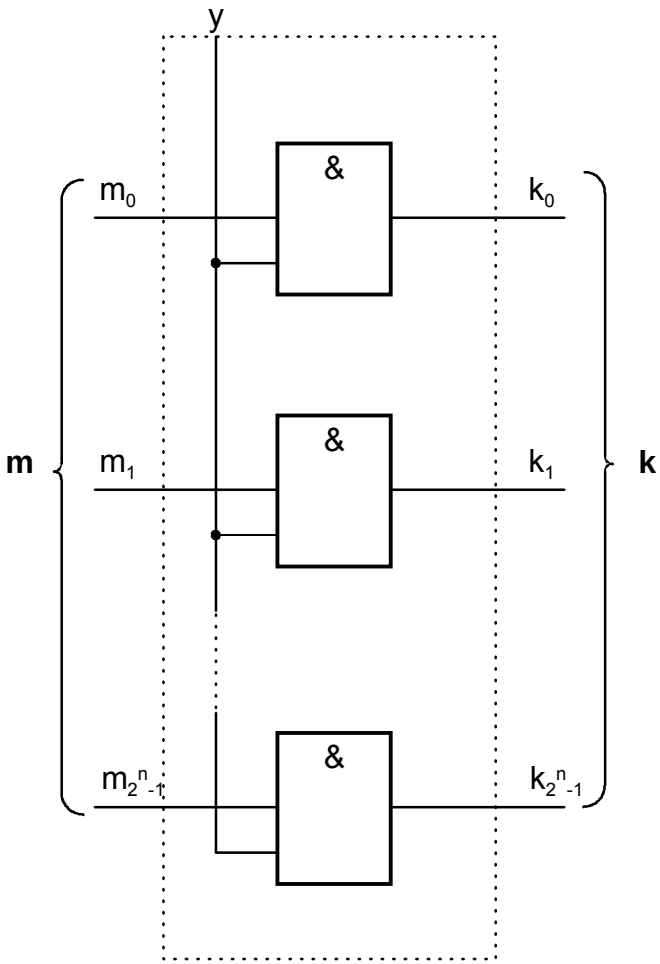


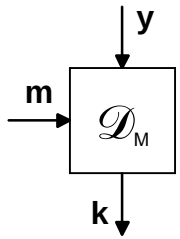
$\mathbf{y} = (\mathbf{x} \,\&\equiv\, \mathbf{W}^{\mathbf{T}})\Sigma\,\&\,\mathbf{K}$; k_j^i = spremenljivke, ki tvorijo vektorje $\mathbf{k}^{\mathbf{i}}$

4. 3. 4 Demultipleksor (Demultiplexer)

4. 3. 4.1 Neadresni skalarni demultipleksor

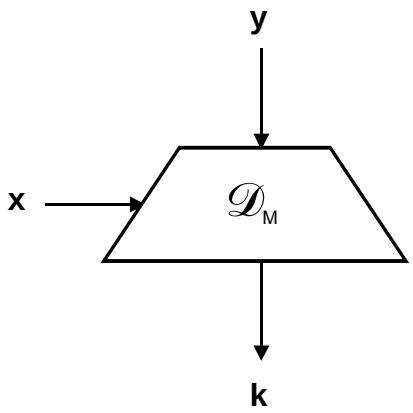
Skalarni neadresni demultipleksor





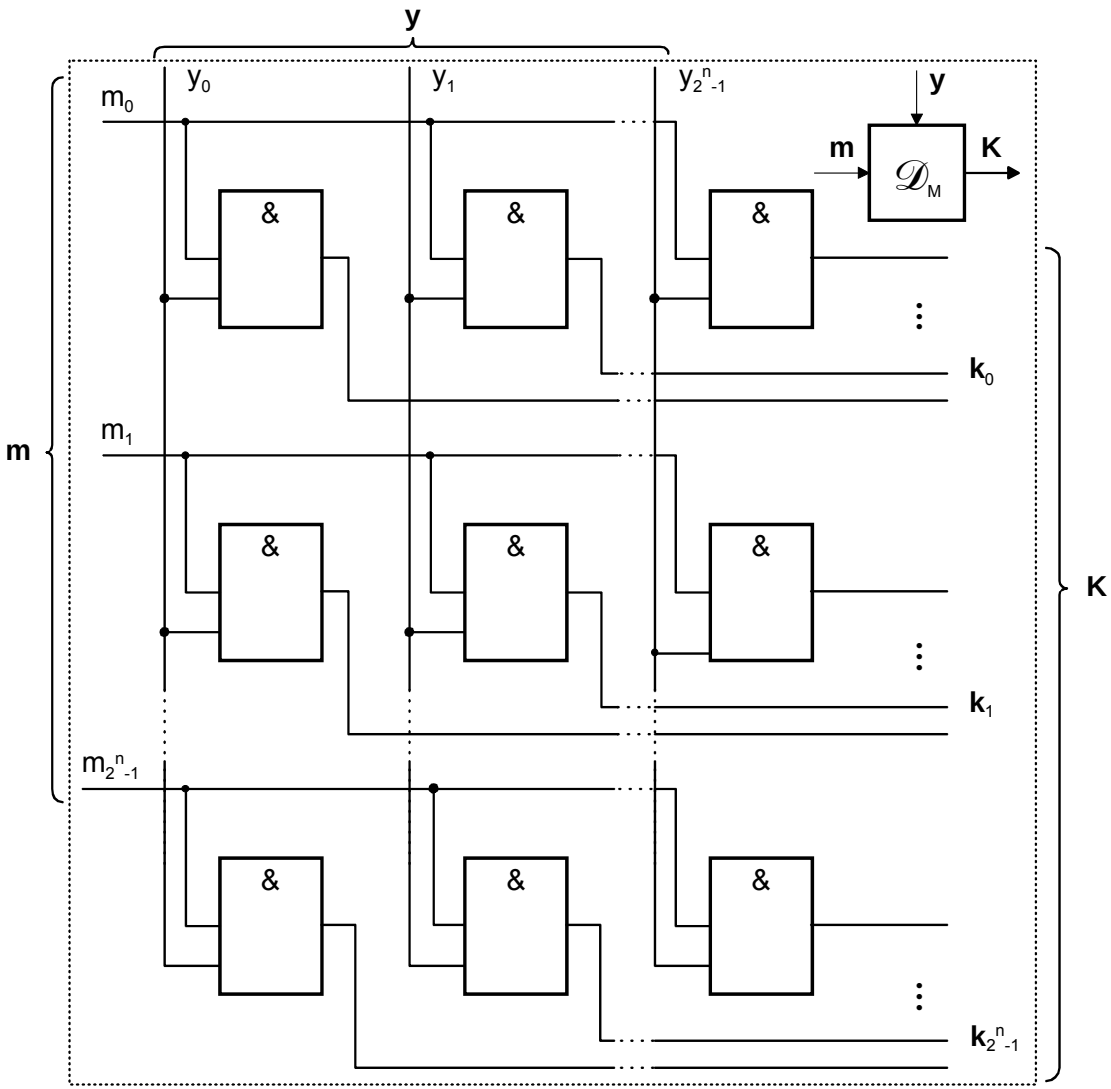
$\mathbf{k} = \mathbf{m}^{T}\Sigma \,\&\, y$ - y je skalarna funkcija

4. 3. 4. 2 Adresni skalarni demultipleksor



$\mathbf{k} = (\mathbf{x}\& \equiv \mathbf{W^T})^T \Sigma \,\&\, y$ - y je še vedno skalarna funkcija

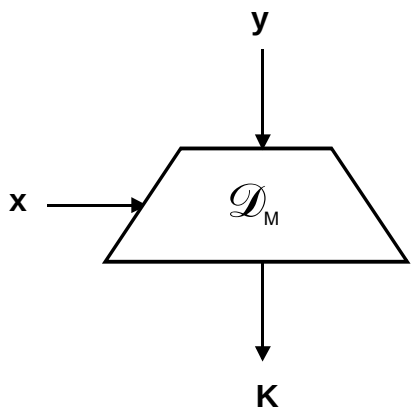
4. 3. 4. 3 Neadresni vektorski demultipleksor



Vsak minterm omogoči nastop ustrezne vrstice matrike **K**, ki predstavlja trenutni izhodni vektor matrike

$$\mathbf{K} = \mathbf{m}^{T} \Sigma \,\&\, \mathbf{y}$$ - **y** je sedaj vektorska vhodna funkcija

4. 3. 4. 4 Adresni vektorski demultipleksor



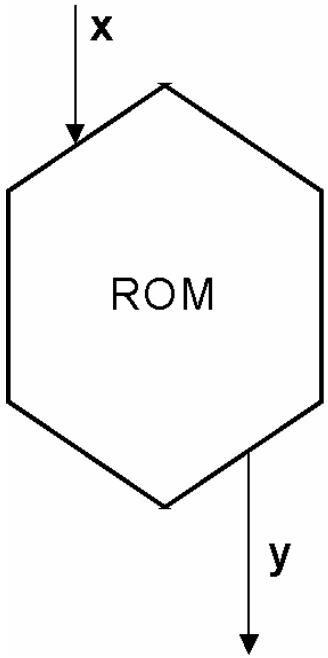
$\mathbf{K} = (x\& \equiv W^T)^T \Sigma \& \mathbf{y}$ - **y** je vektorska vhodna funkcija

4. 3. 5 Programibilna preklopna vezja

$\mathbf{y} = (\mathbf{x} \& \equiv \mathbf{W^T}) \Sigma \& \mathbf{K}$; k_j^i = spremenljivka - skalarna

Tip vezja	Polje - AND	Polje - OR
Programibilni bralni pomnilnik	Fiksno	Programibilno
Programibilne logične mreže - PLA	Programibilno	Programibilno
Programibilna logična polja - PAL	Programibilno	Fiksno

4. 3. 5.1 Bralni pomnilnik (Read Only Memory - ROM)



Naslavljanje pomnilnika:

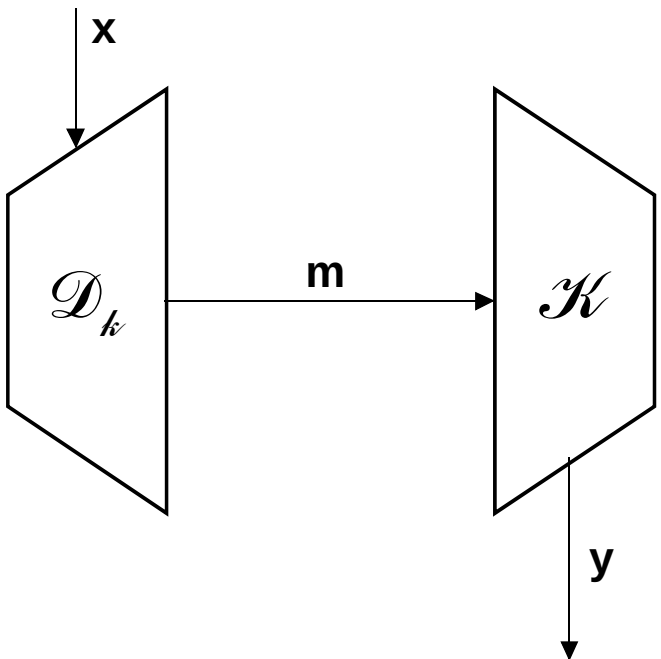
$$\mathbf{m} = \mathbf{x}\,\&\equiv \mathbf{W}^{\mathbf{T}}\ ;\quad w_j^i = \text{konstanta};\quad \mathbf{W} \Rightarrow \mathbf{D};\ w_j^i \Rightarrow d_j^i$$

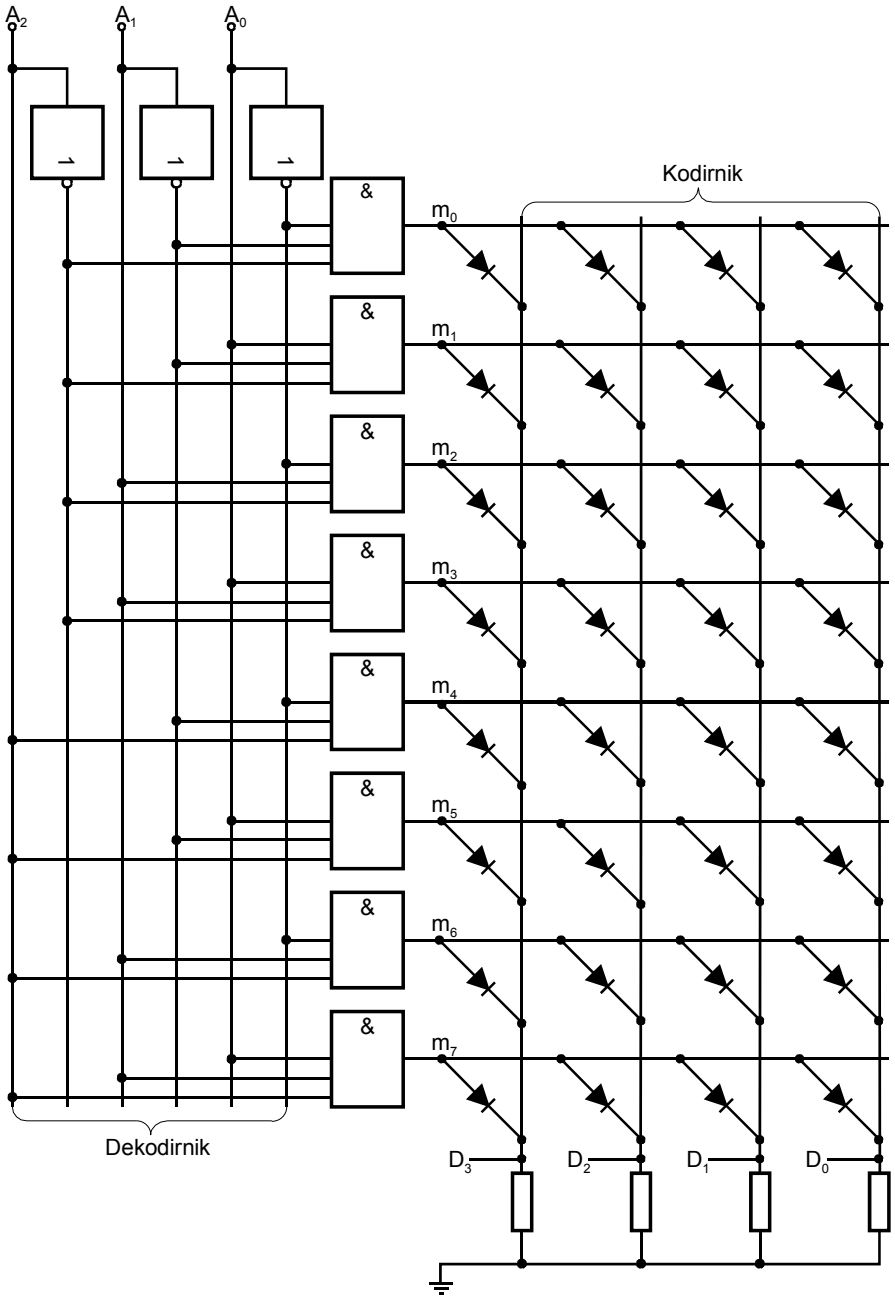
Čitanje pomnilnika

$$\mathbf{y} = \mathbf{m}\,\Sigma\,\&\,\mathbf{K}\ ;\qquad k_j^i = \text{konstanta}$$

Z vstavitvijo prvega izraza v drugega dobimo: popolno disjunktivno normalno obliko vektorske preklopne funkcije – PDNOVPF:

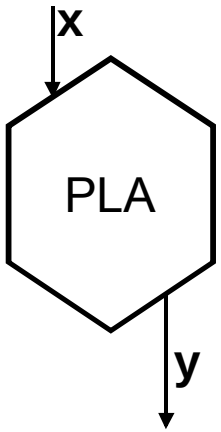
$$\mathbf{y} = (\mathbf{x}\,\&\equiv \mathbf{D}^{\mathbf{T}})\Sigma\,\&\,\mathbf{K}\ ;\ k_j^i = \text{konstanta, ki je bila sprogramirana}$$



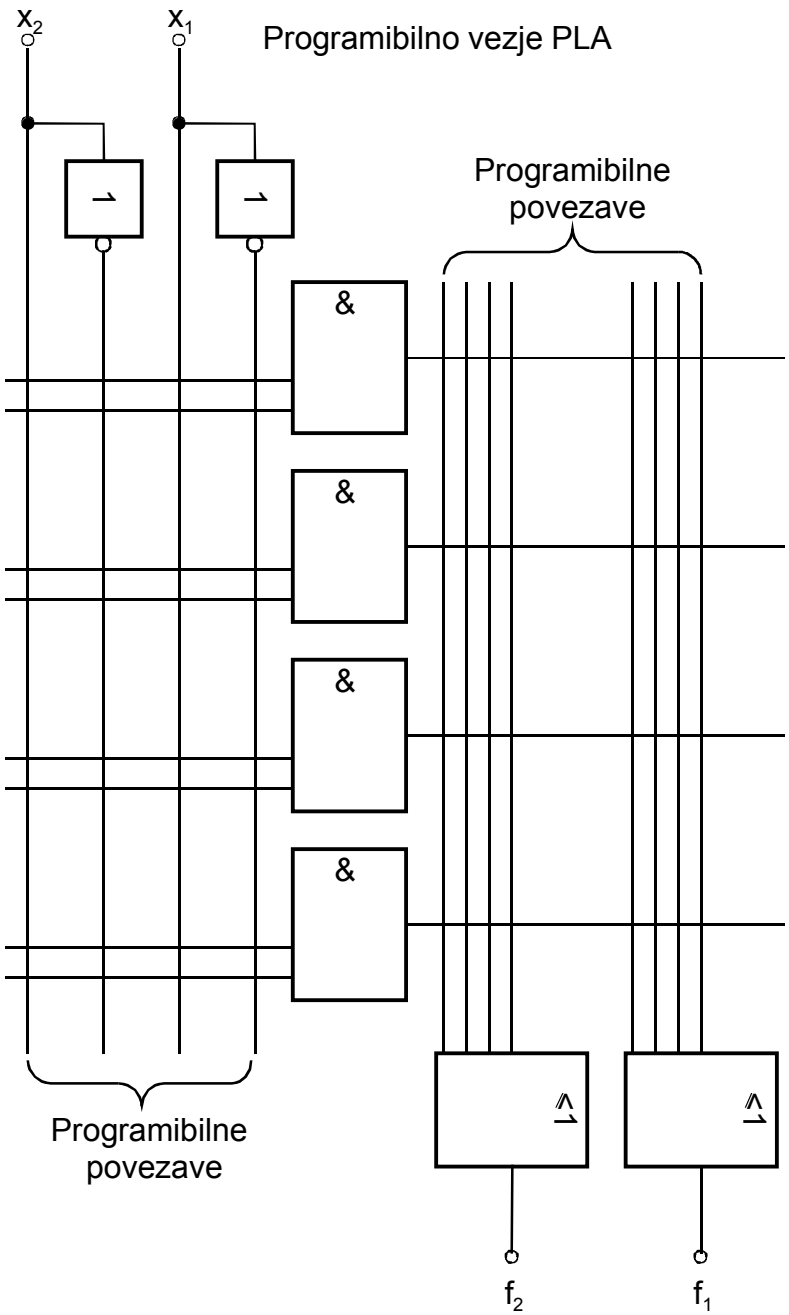

A2
A1
A0
1
1
1
&
&
&
&
&
&
&
&
m0
m1
m2
m3
m4
m5
m6
m7
Kodirnik
Dekodirnik
D3
D2
D1
D0

4. 3. 5. 2 PLA vezja

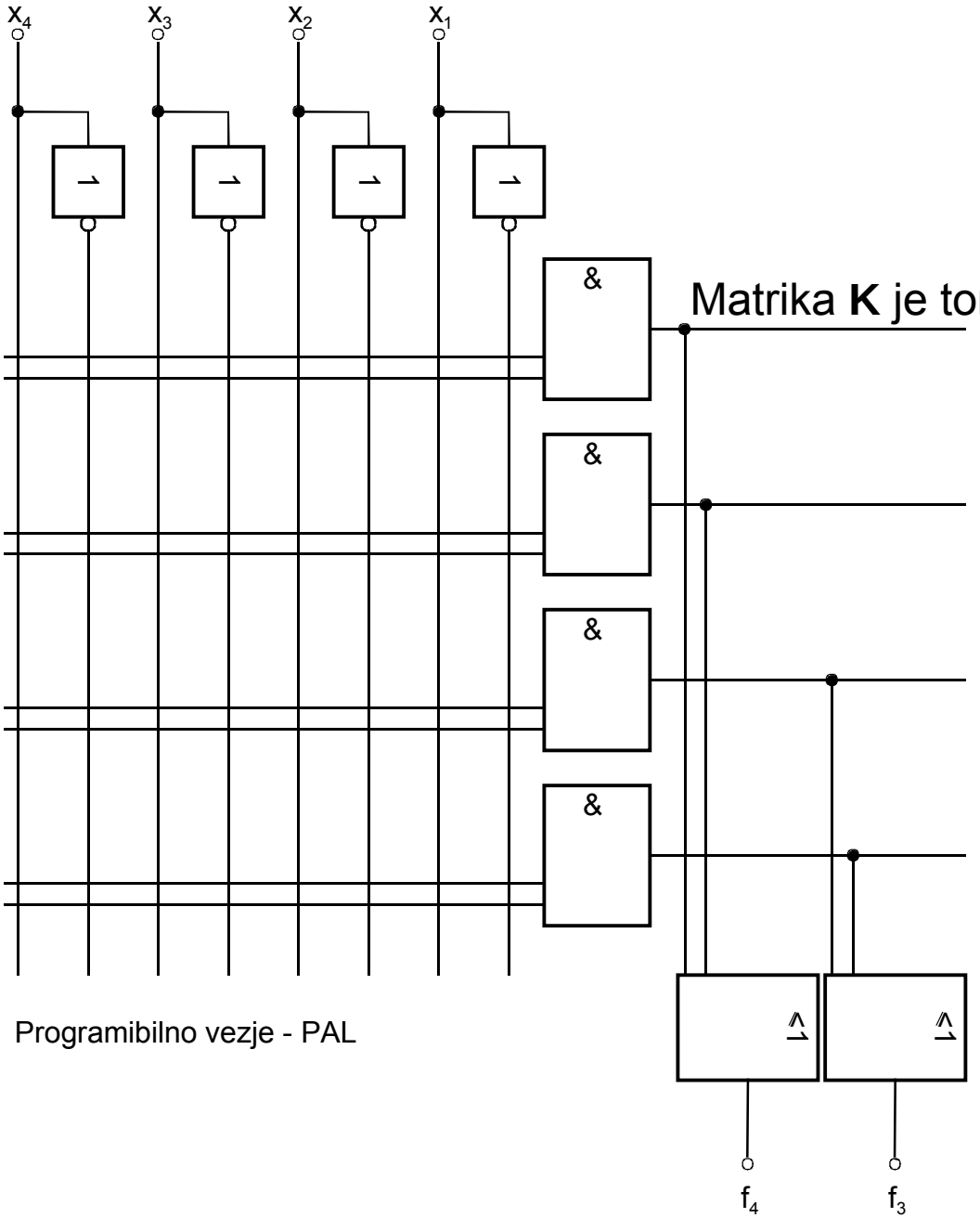
Ta vezja imajo, kot že vemo, programibilni obe polji – dekodirno in kodirno polje.



$$\mathbf{y} = (\mathbf{x} \,\&\, \equiv \mathbf{D}^{\mathbf{T}})\Sigma \,\&\, \mathbf{K}\ ; \qquad k_j^i = \text{konstanta}$$


Programibilno vezje PLA
x2
x1
1
1
Programibilne
povezave
&
&
&
&
=1
=1
Programibilne
povezave
f2
f1

4. 3. 5. 3 PAL vezja:



Matrika **K** je torej določena

Programibilno vezje - PAL

4. 4 Realizacija skalarnih in vektorskih preklopnih funkcij s preklopnimi strukturami

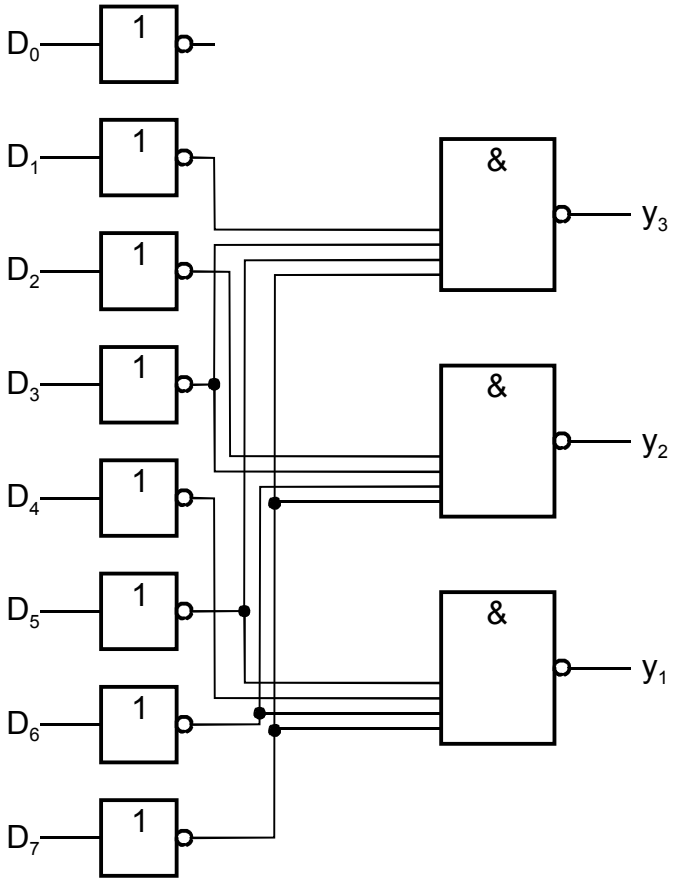
4. 4. 1 Oktalno - binarno kodirno vezje

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	y_1	y_2	y_3
1								0	0	0
	1							0	0	1
		1						0	1	0
			1					0	1	1
				1				1	0	0
					1			1	0	1
						1		1	1	0
							1	1	1	1

$$y_1 = D_4 + D_5 + D_6 + D_7$$

$$y_2 = D_2 + D_3 + D_6 + D_7$$

$$y_3 = D_1 + D_3 + D_5 + D_7$$



4. 4. 2 Kodirnik prioritete – 3 vhodni

P_1 naj bo kodiran z 11

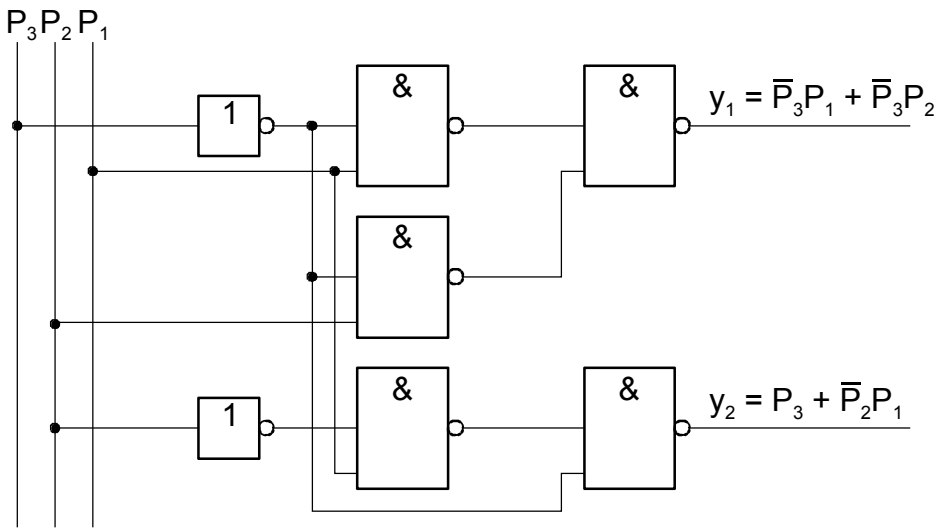
P_2 naj bo kodiran z 10

P_3 naj bo kodiran z 01

P_3 naj ima prioriteto nad P_1 in P_2, P_2 pa nad P_1.

P_3	P_2	P_1	y_1	y_2
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

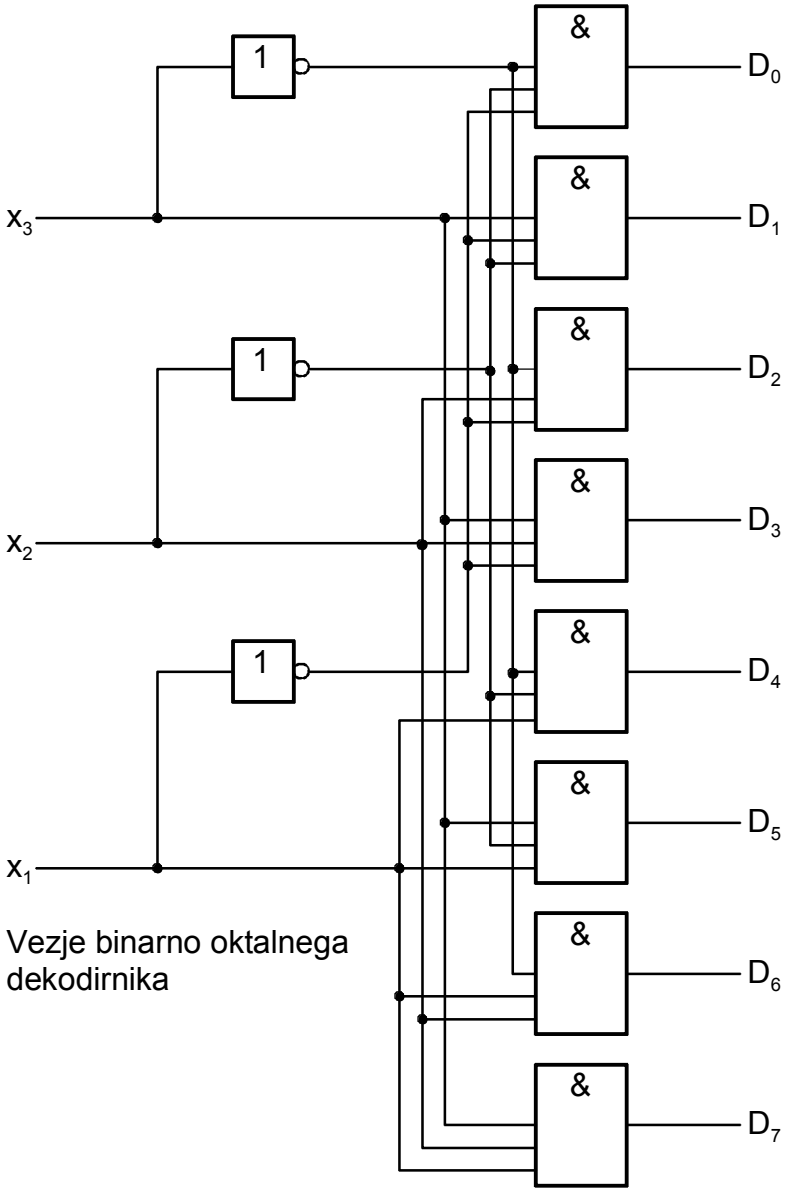
$$y_1 = P_1P_2\overline{P}_3 + P_1\overline{P}_2\overline{P}_3 + \overline{P}_1P_2\overline{P}_3 = P_1\overline{P}_3 + P_2\overline{P}_3 \qquad y_2 = P_1P_2P_3 + \overline{P}_1P_2P_3 + \overline{P}_1\overline{P}_2P_3 + P_1\overline{P}_2P_3 + P_1\overline{P}_2\overline{P}_3 = P_3 + P_1\overline{P}_2$$



4. 4. 3 Binarno oktalni dekodirnik

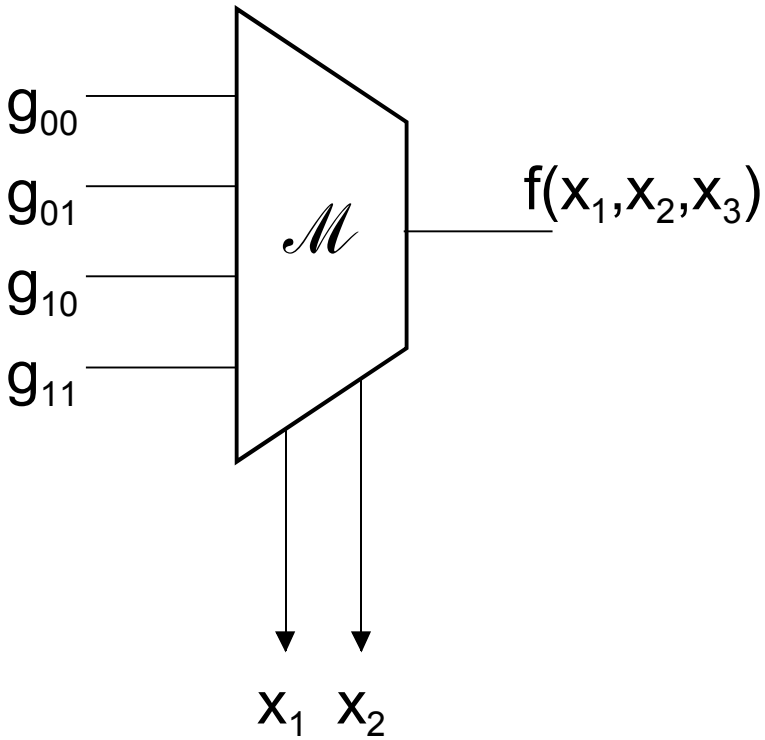
x_1	x_2	x_3	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1							
0	0	1		1						
0	1	0			1					
0	1	1				1				
1	0	0					1			
1	0	1						1		
1	1	0							1	
1	1	1								1

$$D_0 = \overline{x}_1\overline{x}_2\overline{x}_3$$
$$D_1 = \overline{x}_1\overline{x}_2 x_3$$
$$D_2 = \overline{x}_1 x_2\overline{x}_3$$
$$D_3 = \overline{x}_1 x_2 x_3$$
$$D_4 = x_1\overline{x}_2\overline{x}_3$$
$$D_5 = x_1\overline{x}_2 x_3$$
$$D_6 = x_1 x_2\overline{x}_3$$
$$D_7 = x_1 x_2 x_3$$



Vezje binarno oktalnega dekodirnika

4. 4. 4 Multipleksor



x = $\{x_1, x_2, \ldots\ldots\ldots, x_n\}$.

Te spremenljivke pri realizaciji funkcije z multipleksorjem razdelimo v dva dela; v podatkovne in adresne spremenljivke.

Podatkovne in adresne spremenljivke:

$\mathbf{X}_d = \{x_{d1}, x_{d2}, \ldots\ldots\ldots, x_{dd}\} \supset \mathbf{X}$,

$\mathbf{X}_a = \{x_{a1}, x_{a2}, \ldots\ldots\ldots, x_{aa}\}. \supset \mathbf{X}$,

razdelimo na takšen način, da vedno velja:

$\mathbf{X}_a \cup \mathbf{X}_d = \mathbf{X}$

$\mathbf{X}_a \cap \mathbf{X}_d = 0$

Za adresne spremenljivke najprej izberimo »a« zaporednih neodvisnih spremenljivk in naj bo:

a = 2. Torej bo:

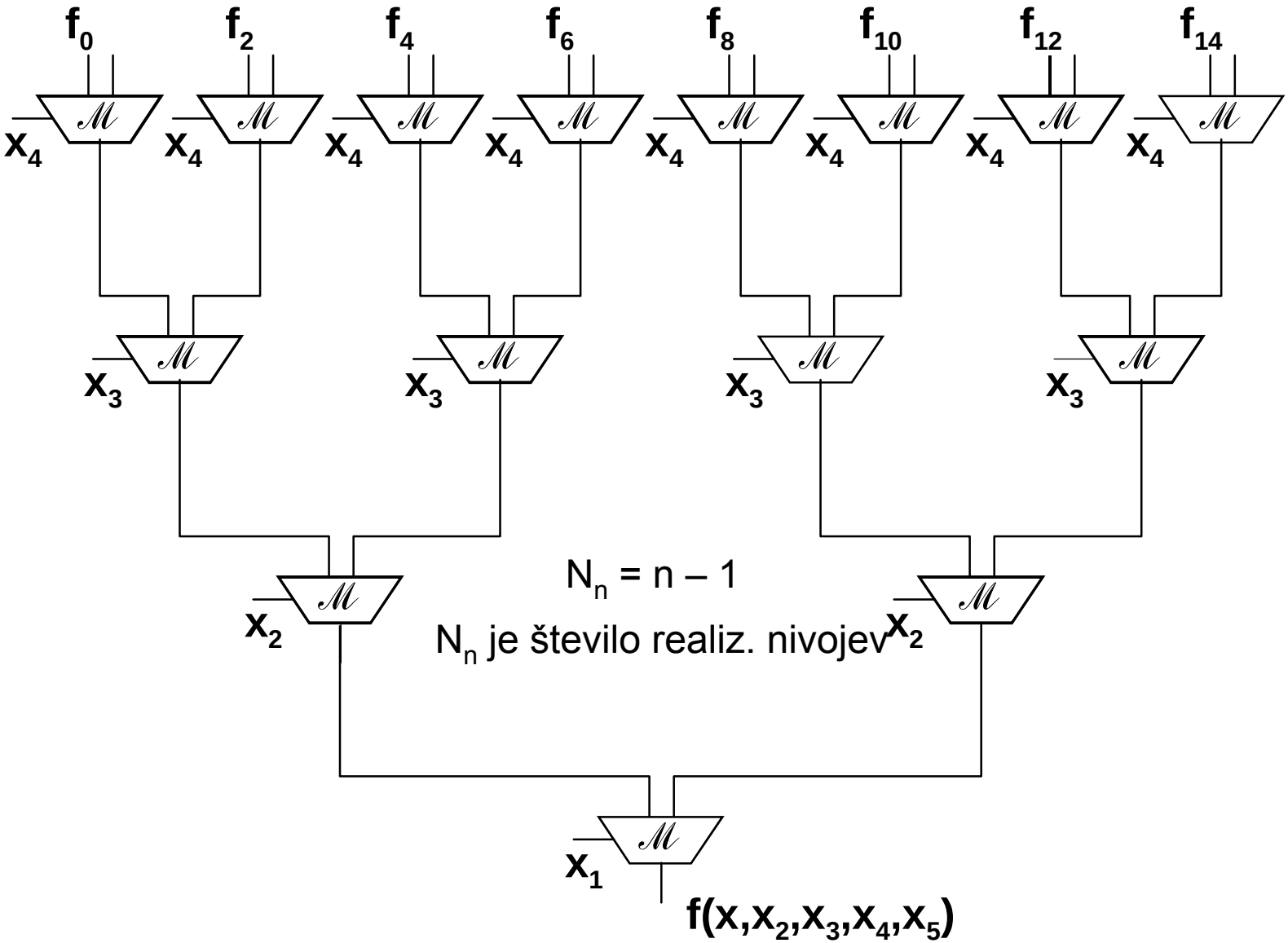
$x_{a1} = x_1$ in $x_{aa} = x_2$.

$f(x_1, x_2, \ldots x_n) = \bar{x}_1 \bar{x}_2 g_{00} + \bar{x}_1 x_2 g_{01} + x_1 \bar{x}_2 g_{10} + x_1 x_2 g_{11}$

kjer je na primer:

$g_{01} = f(0, 1, x_3, \ldots\ldots x_n)$.

Funkcije g_{00}, g_{01}, g_{10}, g_{11} so funkcijski ostanki, ki jih kot vemo, lahko obravnavamo na enak način kot izhodiščno funkcijo, dokler ne začnejo nastopati konstante ali posamične neodvisne spremenljivke.



$N_n = n - 1$

N_n je število realiz. nivojev

$N_M = \sum_{i=0}^{N_n-1} 2^i$; kjer je N_M - število vseh potrebnih multipleksorjev

Pri vseh vmesnih kombinacijah, ko je $1 < a < n - 1$ bo število vseh multipleksorjev odvisno od »a« pri čemer morata biti števili a in N_n celi števili.

$$(n - 1)/a \leq N_n \leq (n + a - 1)/a$$

in

$$N_M = \sum_{i=0}^{N_n - 1} 2^{ai}$$

Da gornja enačba velja tudi za obe skrajnosti, se lahko prepričamo takole:

V prvem primeru je bil $a = n - 1$, celo število iz neenačbe je 1, vsota pa ima tudi samo en člen, to je $2^0 = 1$.

V drugem primeru, ko je $a = 1$ je celo število iz neenačbe $n - 1$ in število vseh multipleksorjev je:

$$N_M = \sum_{i=0}^{N_n - 1} 2^i = \sum_{i=0}^{n-2} 2^i = \sum_{i=0}^{3} 2^i = 2^0 + 2^1 + 2^2 + 2^3 = 1 + 2 + 4 + 8 = 15$$

Kadar pa je $1 < a < n - 1$ pa ni več vseeno kako razporedimo adresne spremenljivke, ker lahko prinese poenostavljanje znaten prihranek vezja.

Število vseh možnih minimizacijskih postopkov je enako številu kombinacij n spremenljivk s po a elementov.

$N_k = \frac{n!}{a!(n-a)!}$; kjer je N_k število vseh možnih kombinacij.

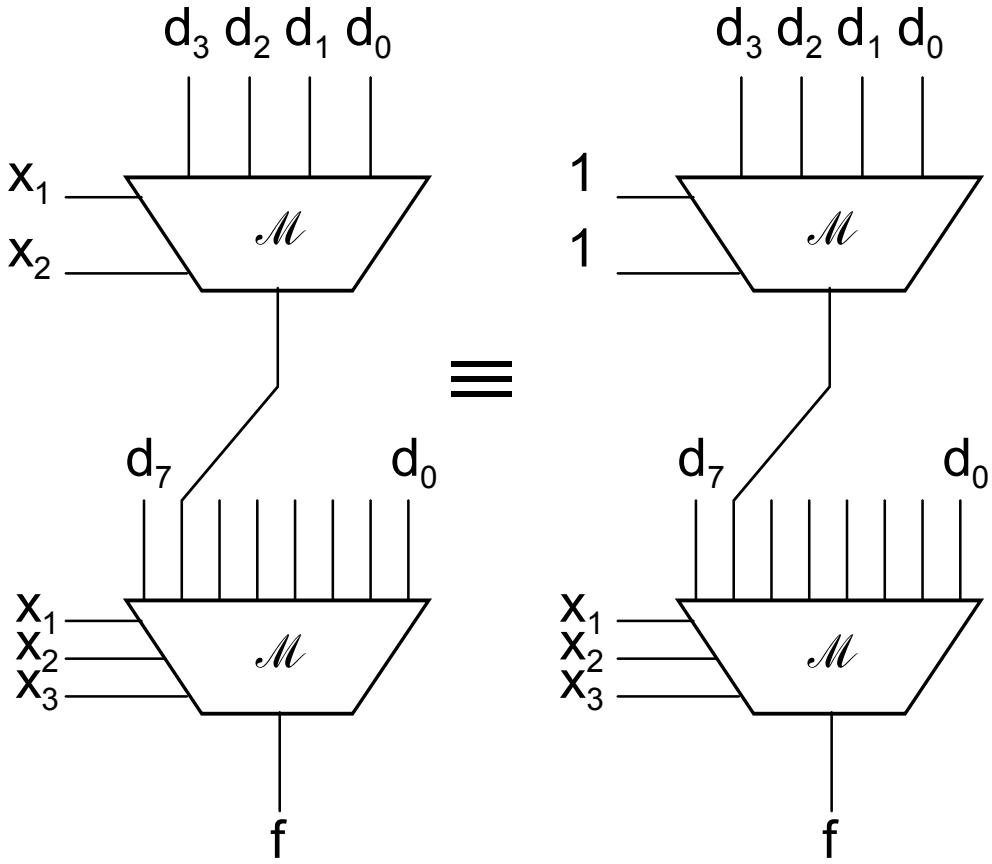
Število Veitchevih diagramov za vsak posamični minimizacijski postopek je potem 2^a.

Vsak posamezen diagram pa ima n-a neodvisnih spremenljivk.

Redundanca pri adresnih spremenljivkah

Kadar število n-1 ni deljivo z a se število adresnih spremenljivk ne ujema s številom adresnih vhodov.

To pa pomeni, da se določene adresne spremnljivke ponovijo na različnih nivojih drevesne strukture.



$$f = \ldots + x_1 x_2 \bar{x}_3 d_6 + \ldots$$

$$f = \ldots + x_1 x_2 \bar{x}_3 (\bar{x}_1 \bar{x}_2 d_0 + \bar{x}_1 x_2 d_1 + x_1 \bar{x}_2 d_2 + x_1 x_2 d_3) + \ldots$$

$$f = \ldots + x_1 x_2 \bar{x}_3 (00 d_0 + 01 d_1 + 10 d_2 + 11 d_3) + \ldots$$

$$f = \ldots + x_1 x_2 \bar{x}_3 (d_3) + \ldots$$

4. 4. 5 Uporaba bralnih pomnilnikov

$$\mathbf{y} = (\mathbf{x} \,\&\equiv \mathbf{D}^{\mathbf{T}})\Sigma \,\& \mathbf{K}\ ; \qquad k_j^i = \text{konstanta}$$

x_1	x_2	x_3	f_1	f_2	f_3	f_4
0	0	0	0	1	1	1
0	0	1	1	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	1	1	0	0	1
1	1	0	1	0	1	1
1	1	1	1	1	1	0

$$f_1 = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3$$

Njena minimalna disjunktivna normalna oblika pa je:

$$f_1 = x_2 + x_3$$

Ker v njem lahko realiziramo le popolno disjunktivno normalno obliko ne moremo izkoristiti minimizacije in s tem prihranka vezja.

Postavlja se vprašanje ali se temu nebi dalo izogniti?

Vzemimo obširnejši primer funkcije s šestimi neodvisnimi spremenljivkami, ki zavzame vrednost 1 pri mintermih z indeksi: 4, 5, 15, 20, 29, 41, 42, 45, 47, 53, 58, 61, 63.

$$f = \sum^{6}(4,5,15,20,29,41,42,45,47,53,58,61,63)$$

Neposredno realizacijo te funkcije bi lahko izvedli na primer 64 x 4 ROM vezjem, oziroma štirimi 16 x 4 ROM vezji.

Naredimo si tabelo nastopajočih mintermov:

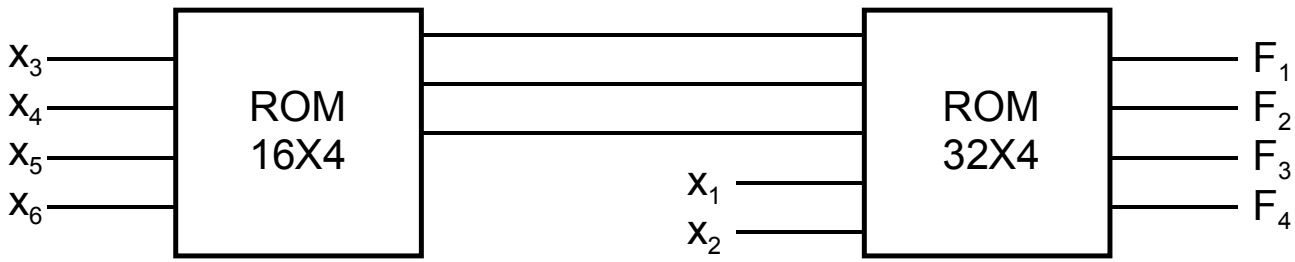
Mintermski indeks	Spremenljivki		Spremenljivke			
	x_1	x_2	x_3	x_4	x_5	x_6
4	0	0	0	1	0	0
5	0	0	0	1	0	1
15	0	0	1	1	1	1
20	0	1	0	1	0	0
29	0	1	1	1	0	1
41	1	0	1	0	0	1
42	1	0	1	0	1	0
45	1	0	1	1	0	1
47	1	0	1	1	1	1
53	1	1	0	1	0	1
58	1	1	1	0	1	0
61	1	1	1	1	0	1
63	1	1	1	1	1	1

Poiščemo vse enake konjunkcije dolžine: $x_3x_4x_5x_6$.

Z njimi naredimo novo tabelo:

spremenljivke				kodirane spremenljivke		
x_3	x_4	x_5	x_6	z_1	z_2	z_3
0	1	0	0	0	0	0
0	1	0	1	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	1

Na drugem nivoju potrebujemo ROM velikosti 32 x 1, kar praktično pomeni 32 x 4.



V tem drugem pomnilniku bo na trinajstih mestih izbranega stolpca zapisana enica, na preostalih devetnajstih pa ničla.

Mesta, kjer mora ostati zapisana enica, so podana v naslednji tabeli; pri tem smo za našo funkcijo rezervirali prvi stolpec, ostali trije pa so še prosti in jih lahko uporabimo za nek drug namen.

Vhodi	Spremenljivke				Izhodi			
x_1	x_2	Z_1	Z_2	Z_3	f_1	f_2	f_3	f_4
0	0	0	0	0	1			
0	0	0	0	1	1			
0	0	1	0	1	1			
0	1	0	0	0	1			
0	1	1	0	0	1			
1	0	0	1	0	1			
1	0	0	1	1	1			
1	0	1	0	0	1			
1	0	1	0	1	1			
1	1	0	0	1	1			
1	1	0	1	1	1			
1	1	1	0	0	1			
1	1	1	0	1	1			

Vzemimo vektorsko funkcijo iz prvega primera.

f_{01}, f_{02}, f_{03}, f_{04}; naj bodo biti besede:

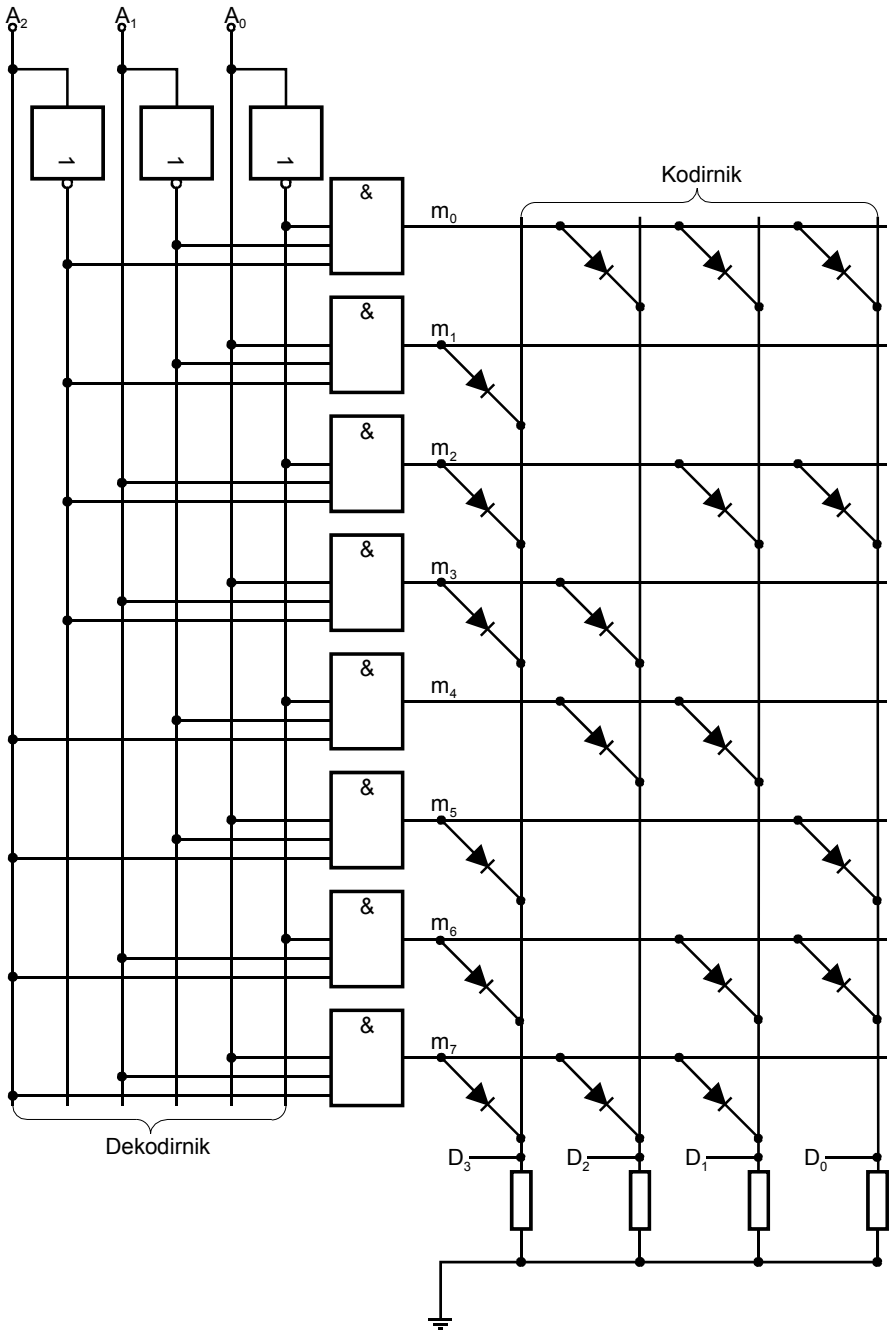
d_{01}, d_{02}, d_{03}, d_{04}, ki jo naslavlja mintrem m_0;

vhodne neodvisne spremenljivke pa preimenujmo v adresne spremenljivke :

A_0, A_1, A_2.

Logično enaka izjavnostna tabela je z novimi oznakami naslednja:

A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	1	1	1
0	0	1	1	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	1	1	0	0	1
1	1	0	1	0	1	1
1	1	1	1	1	1	0


A2
A1
A0
1
1
1
&
&
&
&
&
&
&
&
m0
m1
m2
m3
m4
m5
m6
m7
Kodirnik
Dekodirnik
D3
D2
D1
D0