



Kombinacijska vezja



Kombinacijska vezja

Gradniki nizke, srednje in visoke stopnje integracije

- **kombinacijsko vezje** (*angl. combinational logic circuit*): vezje, katerega izhodi so Booleove funkcije Booleovih vhodnih spremenljivk
- kombinacijska vezja so sestavljena iz logičnih vrat (AND, OR, XOR, NAND, NOR, NXOR, ...), ki smo jih spoznali v prejšnjem poglavju
- integrirana vezja, ki vsebujejo do 10 logičnih vrat (t.j. nekaj deset tranzistorjev), uvrščamo med **gradnike nizke stopnje integracije** (*angl. Small Scale Integration (SSI) circuits*)



Kombinacijska vezja

Gradniki nizke, srednje in visoke stopnje integracije

- v tem poglavju bomo spoznali tipične gradnike **srednje stopnje integracije** (*angl. Medium Scale Integration (MSI)*), ki vsebujejo do 100 logičnih vrat:
 - **kodirnike** in **dekodirnike**,
 - **multipleksorje** in **demultipleksorje**,
 - **primerjalnike** in **seštevalnike**,
 - **aritmetično-logične enote (ALU)**
- v prihodnjem poglavju pa bomo govorili o tipičnih gradnikih **visoke stopnje integracije** (*angl. Large Scale Integration (LSI)*), ki vsebujejo do 10000 vrat; danes so tipični predstavniki preprosta programirljiva vezja (**PROM, EPROM, EEPROM, PAL, PLA, ...**)



Kombinacijska vezja

Gradniki nizke, srednje in visoke stopnje integracije

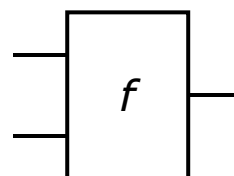
- vezja SSI so bila vrhunec tehnologije od konca 1950-ih do sredine 1960-ih, vezja MSI od konca 1960-ih do začetka 1970-ih, vezja LSI, med katerimi so bili že prvi mikroprocesorji in bralno-pisalni pomnilniki, pa od sredine do konca 1970-ih
- v 1980-ih so sledila vezja **zelo visoke stopnje integracije** (angl. **Very Large Scale Integration (VLSI)** – od 10000 do 10^6 vrat), v 1990-ih pa **ultra visoke stopnje integracije** (angl. **Ultra Large Scale Integration (ULSI)** – več kot 10^6 vrat, redkeje uporabljano)
- danes imajo nekatera vezja že več kot 10^{10} vrat, okrajšave SSI, MSI, LSI oz. VLSI pa uporabljamo le še kot sinonime za vezja z nekaj, nekaj deset, nekaj sto do nekaj tisoč oz. več logičnimi vrati, saj dobesedni pomen npr. pojma "zelo visoka stopnja integracije" ne odraža več trenutnega stanja tehnologije



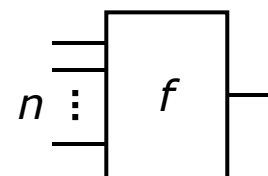
Kombinacijska vezja

Vezja z več vhodi in izhodi

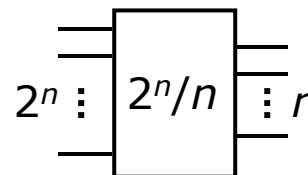
- logična vrata – kombinacijsko vezje z dvema ali več vhodi in enim izhodom:
- kodirniki** in **dekodirniki** – tudi izhodov je več:
- multipleksorji** in **demultipleksorji** – dve vrsti vhodov, **podatkovni** in **izbirni (naslovni)**:



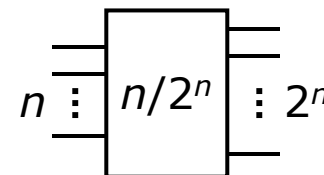
dvovhodna vrata



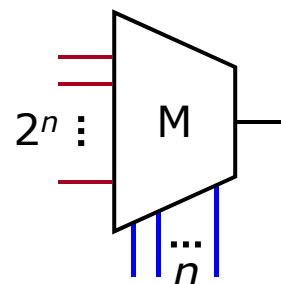
večvhodna vrata



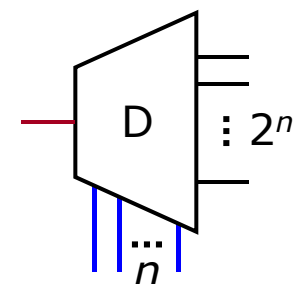
kodirnik



dekodirnik



multipleksor



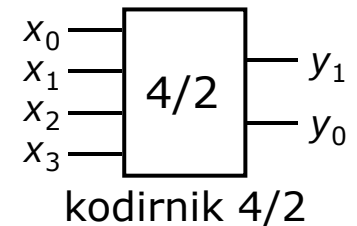
demultipleksor



Kombinacijska vezja

Kodirniki

- **kodirnik** (*angl. encoder*) na izhodih generira kodo (binarno, BCD, ...) naslova (številke) vhoda, na katerem je enica
- najpreprostejši primer: **binarni kodirnik 4/2**



x_0	x_1	x_2	x_3	y_1	y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

to je nepopolna pravilnostna tabela; vezje na njeni osnovi bo delovalo pravilno le, če bo enica vedno na natanko enem vhodu; v tem primeru velja

$$y_1 = x_2 + x_3 ; \quad y_0 = x_1 + x_3$$

- delovanje pri preostalih vhodnih kombinacijah običajno določimo tako, da vhodom pripišemo **prioriteto**; tako dobimo **kodirnik s prioriteto** (*angl. priority encoder*)

x_0	x_1	x_2	x_3	y_1	y_0
0	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

0 ali 1 (ni pomembno)

najvišja prioriteta



Kombinacijska vezja

Kodirniki

x_0	x_1	x_2	x_3	y_1	y_0
0	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$x_2 \backslash x_0 x_1$	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$y_1(x_0, x_1, x_2, x_3) = x_2 + x_3$$

$x_2 \backslash x_0 x_1$	00	01	11	10
00		1	1	
01	1	1	1	1
11	1	1	1	1
10				

$$y_0(x_0, x_1, x_2, x_3) = x_1 \bar{x}_2 + x_3$$

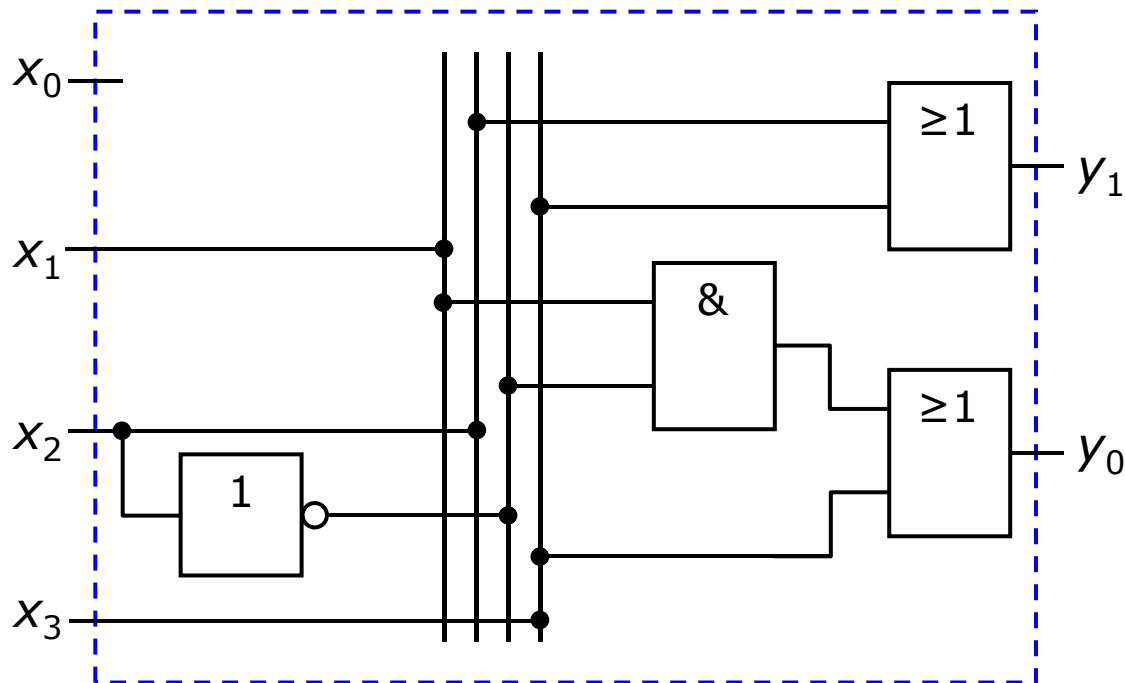


Kombinacijska vezja

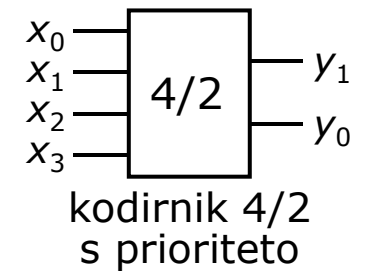
Kodirniki

$$y_1(x_0, x_1, x_2, x_3) = x_2 + x_3$$

$$y_0(x_0, x_1, x_2, x_3) = x_1 \bar{x}_2 + x_3$$



=

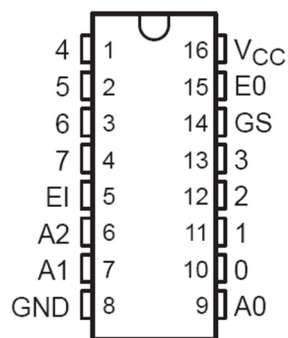
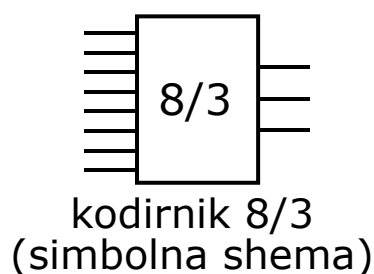


Kombinacijska vezja

Kodirniki

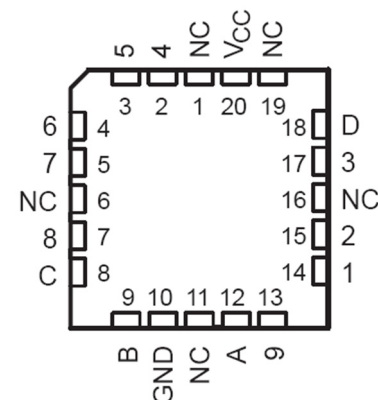
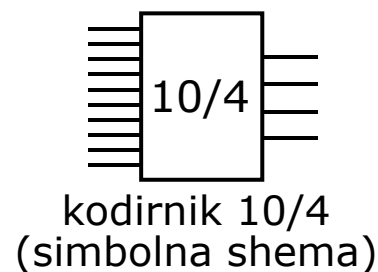
Primeri standardnih kodirnikov:

- **binarni kodirnik 8/3**
($n = 3$, izhod v binarni kodi)



binarni kodirnik 8/3
SN74LS148NS
(Texas Instruments)

- **BCD kodirnik 10/4**
($n = 4$, izhod v BCD kodi, zato le 10 namesto 16 vhodov)



BCD kodirnik 10/4
SN74LS147FK
(Texas Instruments)



Kombinacijska vezja

Dekodirniki

- **dekodirnik** (*angl. decoder*) generira enico na tistem izhodu, katerega naslov (številka) ustreza vrednosti kode (binarne, BCD, ...) na vhodu
- najpreprostejši primer: **binarni dekodirnik 2/4**

x_1	x_0	y_0	y_1	y_2	y_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

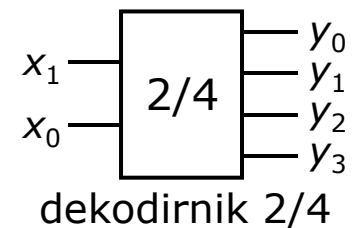
} tokrat je pravilnostna tabela popolna, določanje prioritete zato ni potrebno, iz tabele pa hitro razberemo:

$$y_0 = \bar{x}_1 \bar{x}_0$$

$$y_1 = \bar{x}_1 x_0$$

$$y_2 = x_1 \bar{x}_0$$

$$y_3 = x_1 x_0$$





Kombinacijska vezja

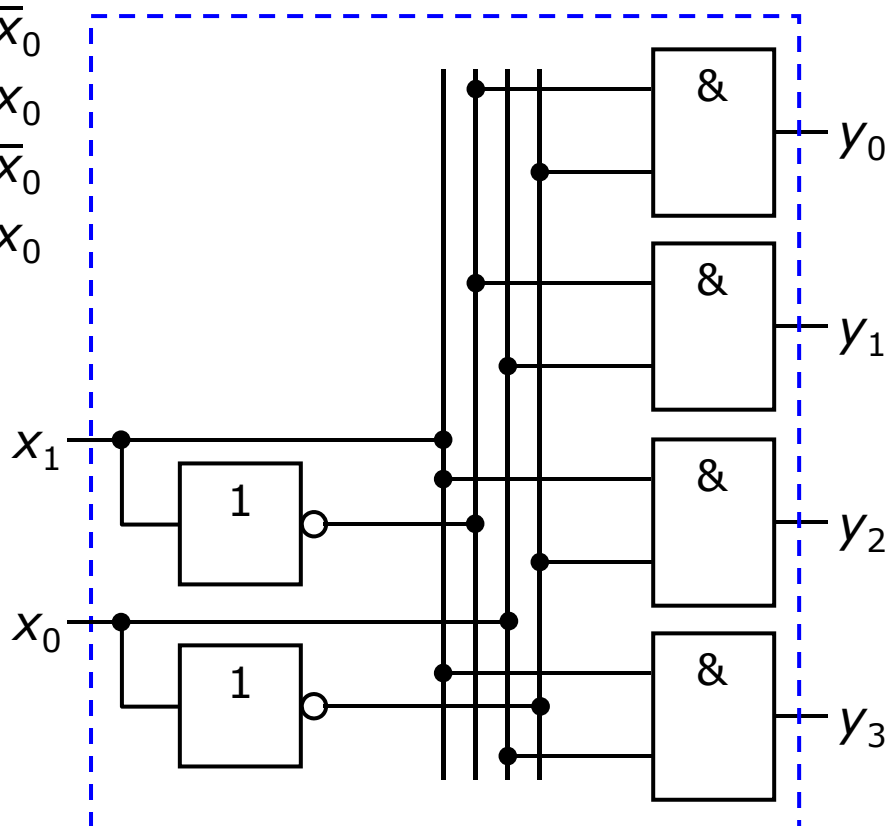
Dekodirniki

$$y_0 = \bar{x}_1 \bar{x}_0$$

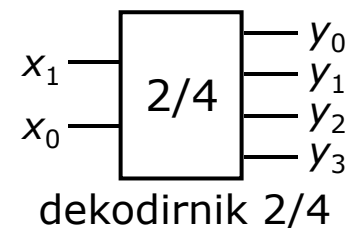
$$y_1 = \bar{x}_1 x_0$$

$$y_2 = x_1 \bar{x}_0$$

$$y_3 = x_1 x_0$$



=



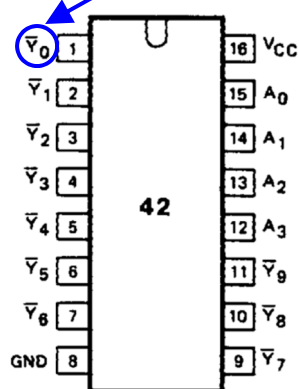
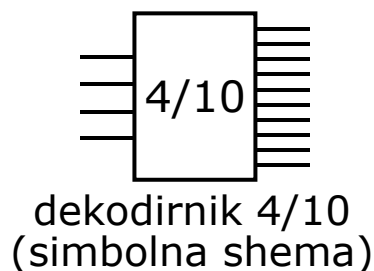


Kombinacijska vezja

Dekodirniki

Primeri standardnih dekodirnikov:

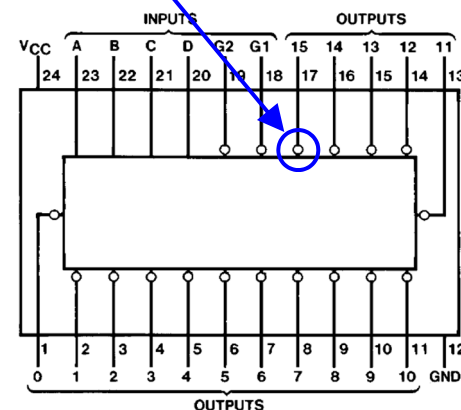
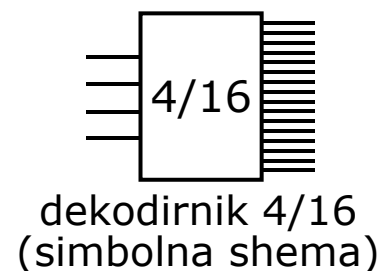
- **dekodirnik 4/10**
(iz BCD kode)



BCD dekodirnik 4/10
74HC42
(Philips Semiconductors)

negacija izhoda pove, da dekodirnik
na aktivnem izhodu generira ničlo,
na neaktivnih pa enice

- **dekodirnik 4/16**
(iz binarne kode)



binarni dekodirnik 4/16
DM74154N
(National Semiconductor)



Kombinacijska vezja

Pozitivna in negativna logika

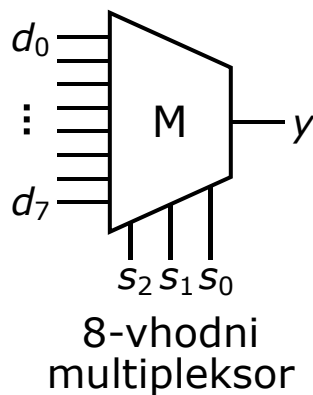
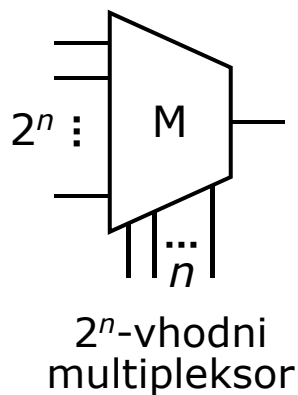
- če v integriranem vezju logični enici ustreza višja, logični ničli pa nižja napetost (npr. 1 = +5V, 0 = 0V), pravimo, da gre za **vezje s pozitivno logiko**
- če v integriranem vezju logični enici ustreza nižja, logični ničli pa višja napetost (npr. 1 = 0V, 0 = +5V), pa gre za **vezje z negativno logiko**
- poglavitni vzrok za uporabo negativne logike je možnost gradnje vezja z manjšim številom vrat in/ali s hitrejšimi vrati
- prototipa kodirnika 4/2 in dekodirnika 2/4, ki smo ju obravnavali, smo zapisali v DNO in izvedli z vrati AND in OR; dejanske izvedbe pretežno uporabljajo vrata NAND in/ali NOR, ki tudi pogosto delujejo hitreje v negativni logiki kot v pozitivni



Kombinacijska vezja

Multiplesorji

- **multiplesor** (*angl. multiplexer, MUX*) ima 2^n podatkovnih in n izbirnih vhodov ter en izhod
- na izhodu je vrednost tistega podatkovnega vhoda (Booleove spremenljivke), katerega naslov je zapisan na izbirnih vhodih



s_2	s_1	s_0	y
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3
1	0	0	d_4
1	0	1	d_5
1	1	0	d_6
1	1	1	d_7

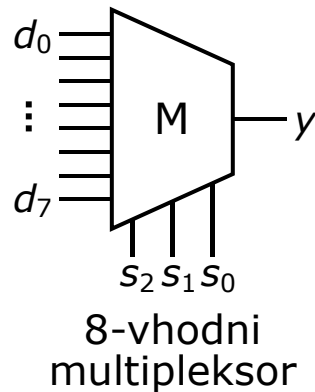
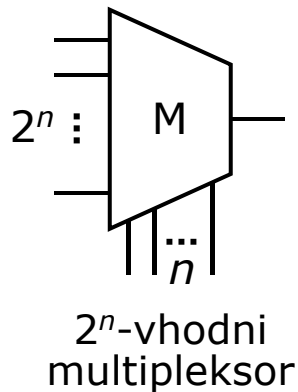
$$\begin{aligned} y &= \bar{s}_2 \bar{s}_1 \bar{s}_0 d_0 + \bar{s}_2 \bar{s}_1 s_0 d_1 \\ &+ \bar{s}_2 s_1 \bar{s}_0 d_2 + \bar{s}_2 s_1 s_0 d_3 \\ &+ s_2 \bar{s}_1 \bar{s}_0 d_4 + s_2 \bar{s}_1 s_0 d_5 \\ &+ s_2 s_1 \bar{s}_0 d_6 + s_2 s_1 s_0 d_7 \\ &= m_0 d_0 + m_1 d_1 + \dots + m_7 d_7 \end{aligned}$$



Kombinacijska vezja

Multipleksorji

- **multipleksor** (*angl. multiplexer, MUX*) ima 2^n podatkovnih in n izbirnih vhodov ter en izhod
- na izhodu je vrednost tistega podatkovnega vhoda (Booleove spremenljivke), katerega naslov je zapisan na izbirnih vhodih



s_2	s_1	s_0	y
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3
1	0	0	d_4
1	0	1	d_5
1	1	0	d_6
1	1	1	d_7

$$\begin{aligned}
 y &= \bar{s}_2\bar{s}_1\bar{s}_0d_0 + \bar{s}_2\bar{s}_1s_0d_1 \\
 &+ \bar{s}_2s_1\bar{s}_0d_2 + \bar{s}_2s_1s_0d_3 \\
 &+ s_2\bar{s}_1\bar{s}_0d_4 + s_2\bar{s}_1s_0d_5 \\
 &+ s_2s_1\bar{s}_0d_6 + s_2s_1s_0d_7 \\
 &= m_0d_0 + m_1d_1 + \dots + m_7d_7
 \end{aligned}$$

$$y = \underline{m} \cdot \underline{d}$$

vektorska
enačba
multipleksorja



Kombinacijska vezja

Multipleksorji

s_1	s_0	y
0	0	d_0
0	1	d_1
1	0	d_2
1	1	d_3

$$\begin{aligned} y &= \bar{s}_1 \bar{s}_0 d_0 + \bar{s}_1 s_0 d_1 + s_1 \bar{s}_0 d_2 + s_1 s_0 d_3 \quad (\text{primerno za izvedbo AND-OR}) \\ &= \overline{\bar{s}_1 \bar{s}_0 d_0 + \bar{s}_1 s_0 d_1 + s_1 \bar{s}_0 d_2 + s_1 s_0 d_3} \\ &= \overline{\bar{s}_1 \bar{s}_0 d_0} \cdot \overline{\bar{s}_1 s_0 d_1} \cdot \overline{s_1 \bar{s}_0 d_2} \cdot \overline{s_1 s_0 d_3} \\ &= \uparrow(\uparrow(\bar{s}_1, \bar{s}_0, d_0), \uparrow(\bar{s}_1, s_0, d_1), \uparrow(s_1, \bar{s}_0, d_2), \uparrow(s_1, s_0, d_3)) \\ &\quad (\text{primerno za izvedbo NAND-NAND}) \end{aligned}$$

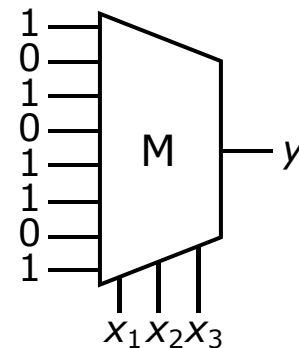
- za vajo izrišite simbolne sheme 4-vhodnega multipleksorja:
 - v izvedbi z negatorji ter vrati AND in OR
 - v izvedbi z negatorji ter vrati NAND

Kombinacijska vezja

Uporaba multipleksorja za realizacijo preklapne funkcije

- z multipleksorjem lahko realiziramo poljubno preklapno funkcijo:
 - trivialna realizacija:** za $y = f(x_1, x_2, \dots, x_n)$ uporabimo 2^n -vhodni multipleksor; na izbirne vhode pripeljemo spremenljivke, na podatkovne pa konstante iz izhodnega stolpca pravilnostne tabele

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



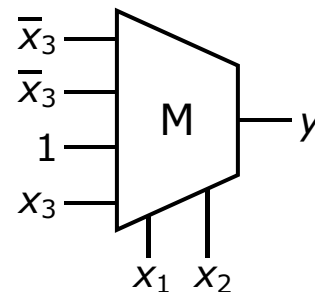
Kombinacijska vezja

Uporaba multipleksorja za realizacijo preklapne funkcije

- z multipleksorjem lahko realiziramo poljubno preklapno funkcijo:
 - netrivialna realizacija:** uporabimo manjši multipleksor;
na izbirne vhode pripeljemo nekaj spremenljivk, na podatkovne pa preostale spremenljivke, njihove funkcije in/ali konstante

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$\begin{aligned}
 y &= \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2x_3 \\
 &= \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2(\bar{x}_3 + x_3) + x_1x_2x_3 \\
 &= \bar{x}_1\bar{x}_2(\bar{x}_3) + \bar{x}_1x_2(\bar{x}_3) + x_1\bar{x}_2(1) + x_1x_2(x_3)
 \end{aligned}$$



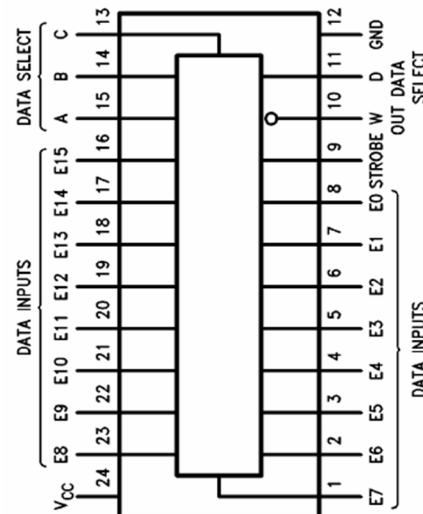
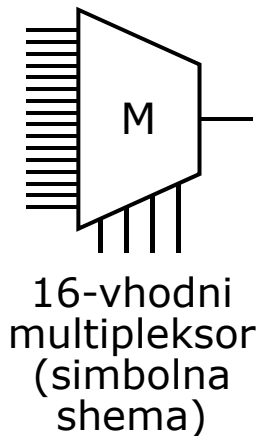


Kombinacijska vezja

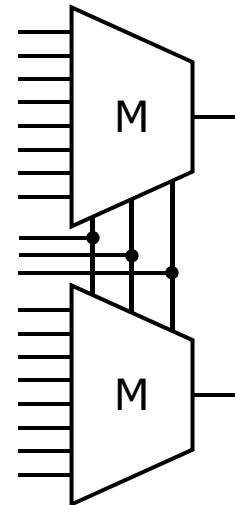
Multiplesorji

Primera standardnih multiplesorjev:

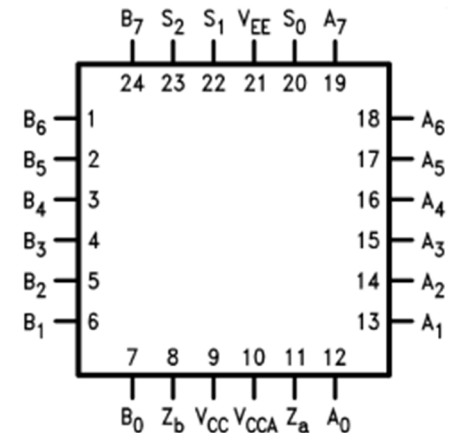
- **16-vhodni multiplesor**
- **dvojni 8-vhodni multiplesor**
(dva 8-vhodna, skupni izbirni vh.)



16-vhodni MUX
DM74150N
(National
Semiconductor)



dvojni 8-vhodni
multiplesor
(simbolna shema)

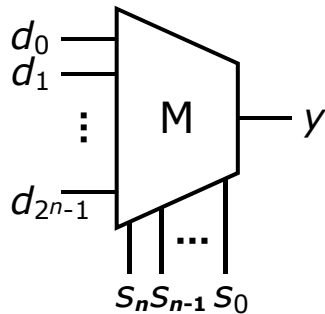


dvojni 8-vhodni MUX
100363-CPK
(National
Semiconductor)

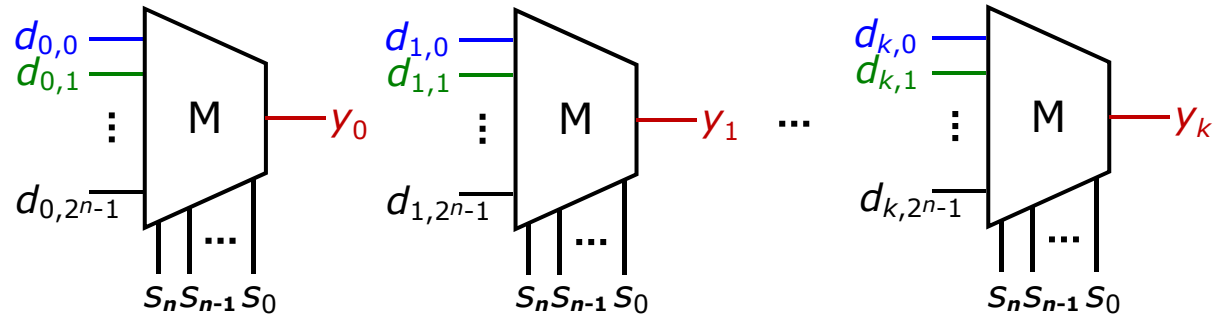
Kombinacijska vezja

Multiplesorji

- dvojni multiplesor je najpreprostejši primer splošnejšega **vektorskega multiplesorja**; navadnemu multiplesorju v takšni terminologiji pravimo **skalarni multiplesor**



2^n -vhodni
skalarni
multiplesor



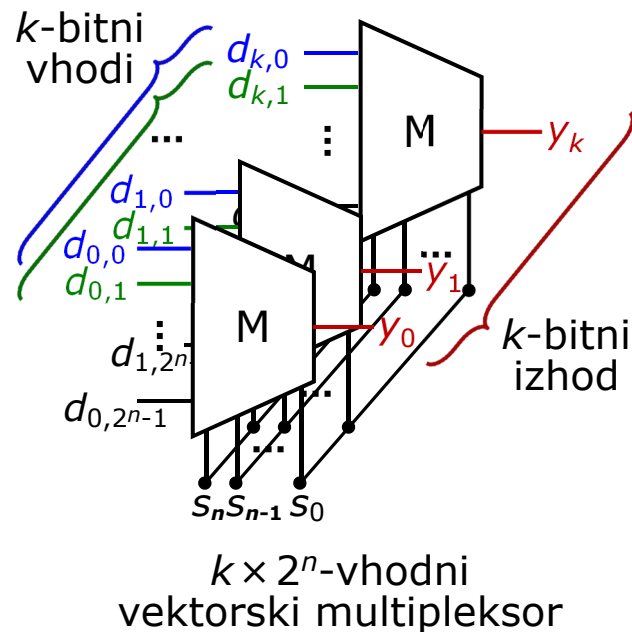
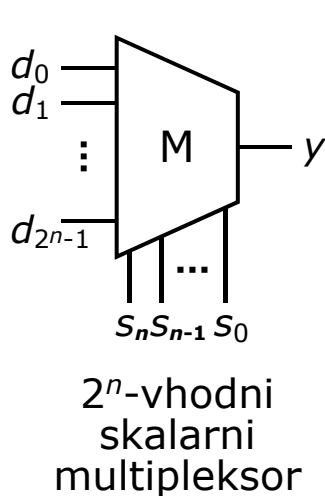
k 2^n -vhodnih skalarnih multiplesorjev
s skupnimi izbirnimi vhodi



Kombinacijska vezja

Multipleksorji

- dvojni multipleksor je najpreprostejši primer splošnejšega **vektorskega multipleksorja**; navadnemu multipleksorju v takšni terminologiji pravimo **skalarni multipleksor**



S_n	S_{n-1}	...	S_0	y_0	y_1	...	y_k
0	0	...	0	$d_{0,0}$	$d_{1,0}$	...	$d_{k,0}$
0	0	...	1	$d_{0,1}$	$d_{1,1}$	...	$d_{k,1}$
0	0	...	0	$d_{0,2}$	$d_{1,2}$	...	$d_{k,2}$
...	...	...	...	...	...	...	...
1	1	...	1	$d_{0,2^n-1}$	$d_{1,2^n-1}$	...	$d_{k,2^n-1}$



Kombinacijska vezja

Multipleksorji

- dvojni multipleksor je najpreprostejši primer splošnejšega **vektorskega multipleksorja**; navadnemu multipleksorju v takšni terminologiji pravimo **skalarni multipleksor**

S_n	S_{n-1}	...	S_0	y_0	y_1	...	y_k
0	0	...	0	$d_{0,0}$	$d_{1,0}$	...	$d_{k,0}$
0	0	...	1	$d_{0,1}$	$d_{1,1}$	...	$d_{k,1}$
0	0	...	0	$d_{0,2}$	$d_{1,2}$	...	$d_{k,2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
1	1	...	1	$d_{0,2^n-1}$	$d_{1,2^n-1}$	...	$d_{k,2^n-1}$

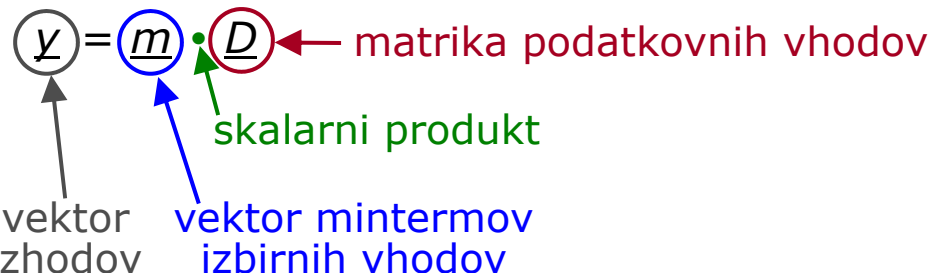
$$y_0 = m_0 d_{0,0} + m_1 d_{0,1} + \dots + m_{2^n-1} d_{0,2^n-1}$$

$$y_1 = m_0 d_{1,0} + m_1 d_{1,1} + \dots + m_{2^n-1} d_{1,2^n-1}$$

$$y_2 = m_0 d_{2,0} + m_1 d_{2,1} + \dots + m_{2^n-1} d_{2,2^n-1}$$

$$\vdots$$

$$y_k = m_0 d_{k,0} + m_1 d_{k,1} + \dots + m_{2^n-1} d_{k,2^n-1}$$





Kombinacijska vezja

Multipleksorji

- dvojni multipleksor je najpreprostejši primer splošnejšega **vektorskega multipleksorja**; navadnemu multipleksorju v takšni terminologiji pravimo **skalarni multipleksor**

s_n	s_{n-1}	...	s_0	y_0	y_1	...	y_k
0	0	...	0	$d_{0,0}$	$d_{1,0}$	...	$d_{k,0}$
0	0	...	1	$d_{0,1}$	$d_{1,1}$	...	$d_{k,1}$
0	0	...	0	$d_{0,2}$	$d_{1,2}$	...	$d_{k,2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
1	1	...	1	$d_{0,2^n-1}$	$d_{1,2^n-1}$	...	$d_{k,2^n-1}$

$$y_0 = m_0 d_{0,0} + m_1 d_{0,1} + \dots + m_{2^n-1} d_{0,2^n-1}$$

$$y_1 = m_0 d_{1,0} + m_1 d_{1,1} + \dots + m_{2^n-1} d_{1,2^n-1}$$

$$y_2 = m_0 d_{2,0} + m_1 d_{2,1} + \dots + m_{2^n-1} d_{2,2^n-1}$$

$$\vdots$$

$$y_k = m_0 d_{k,0} + m_1 d_{k,1} + \dots + m_{2^n-1} d_{k,2^n-1}$$

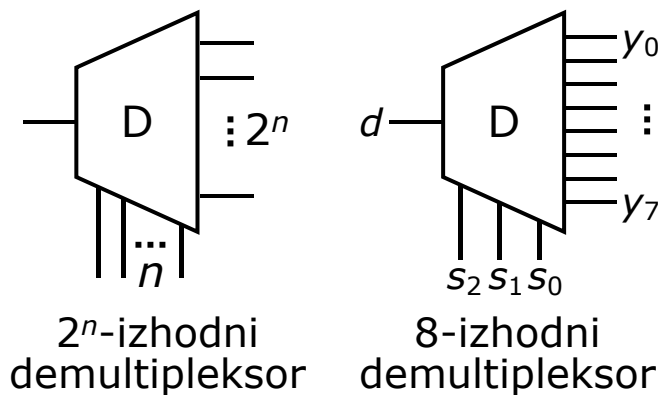
$$\underline{y} = \underline{m} \cdot \underline{D}$$

matrična
enačba
vektorskega
multipleksorja

Kombinacijska vezja

Demultipleksorji

- **demultipleksor** (*angl. demultiplexer*) ima en podatkovni in n izbirnih vhodov ter 2^n izhodov
- vrednost podatkovnega vhoda se prenese na tisti izhod, katerega naslov je zapisan na izbirnih vhodih



s_2	s_1	s_0	y_0	y_1	...	y_7
0	0	0	d	0	...	0
0	0	1	0	d	...	0
0	1	0	0	0	...	0
0	1	1	0	0	...	0
1	0	0	0	0	...	0
1	0	1	0	0	...	0
1	1	0	0	0	...	0
1	1	1	0	0	...	d

$$y_0 = \bar{s}_2 \bar{s}_1 \bar{s}_0 d = m_0 d$$

$$y_1 = \bar{s}_2 \bar{s}_1 s_0 d = m_1 d$$

$$\vdots$$

$$y_7 = s_2 s_1 s_0 d = m_7 d$$

vektor izhodov

podatkovni vhod (skalar)

transponirani vektor min-termov izbirnih vhodov

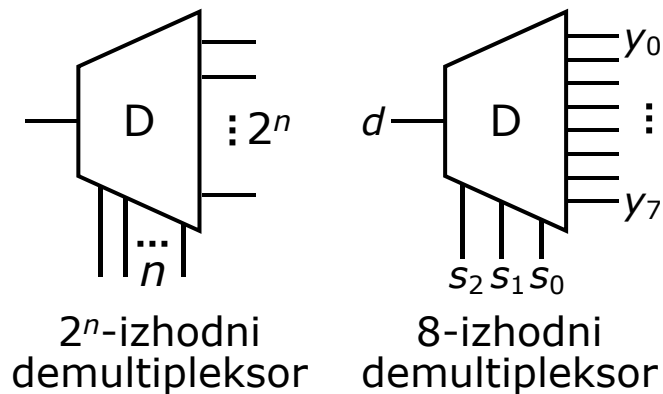
vsota produktov (skalarni produkt)

$$\underline{y} = \underline{m}^T \cdot d$$

Kombinacijska vezja

Demultipleksorji

- **demultipleksor** (*angl. demultiplexer*) ima en podatkovni in n izbirnih vhodov ter 2^n izhodov
- vrednost podatkovnega vhoda se prenese na tisti izhod, katerega naslov je zapisan na izbirnih vhodih



s_2	s_1	s_0	y_0	y_1	...	y_7
0	0	0	d	0	...	0
0	0	1	0	d	...	0
0	1	0	0	0	...	0
0	1	1	0	0	...	0
1	0	0	0	0	...	0
1	0	1	0	0	...	0
1	1	0	0	0	...	0
1	1	1	0	0	...	d

$$y_0 = \bar{s}_2 \bar{s}_1 \bar{s}_0 d = m_0 d$$

$$y_1 = \bar{s}_2 \bar{s}_1 s_0 d = m_1 d$$

$$\vdots$$

$$y_7 = s_2 s_1 s_0 d = m_7 d$$

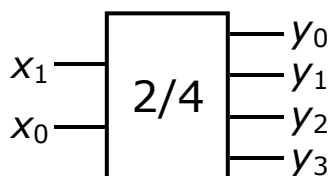
$$\underline{y} = \underline{m}^T \cdot d$$

vektorska enačba
(skalarne)
demultipleksorja

Kombinacijska vezja

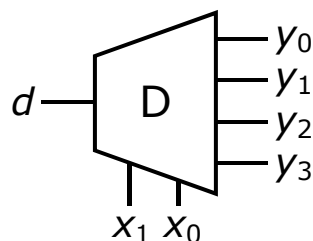
Demultipleksorji

- če 2^n -izhodnemu demultipleksorju na podatkovni vhod vežemo enico, dobimo dekodirnik $n/2^n$



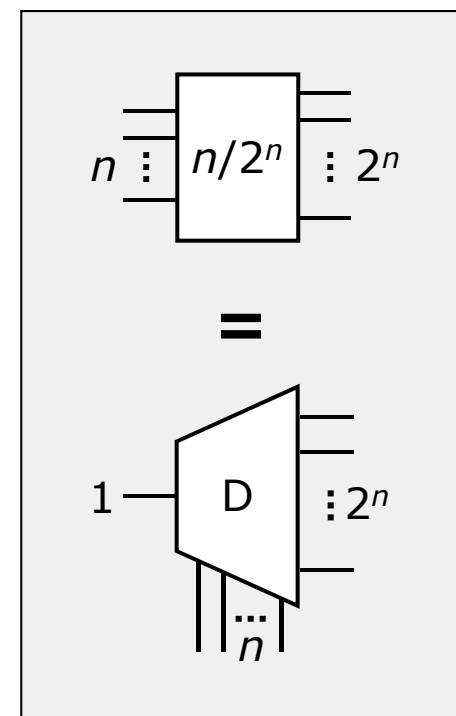
x_1	x_0	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

dekodirnik 2/4 s
pravilnostno tabelo



x_1	x_0	y_0	y_1	y_2	y_3
0	0	d	0	0	0
0	1	0	d	0	0
1	0	0	0	d	0
1	1	0	0	0	d

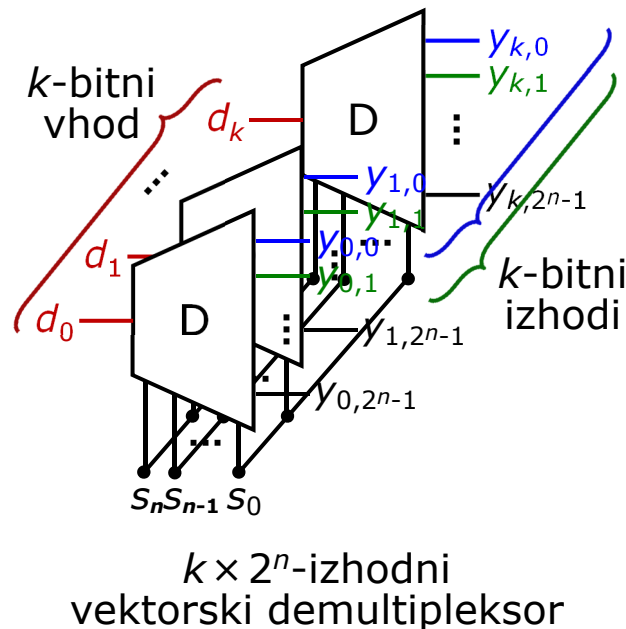
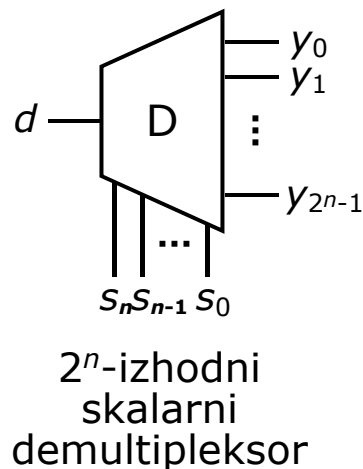
4-izh. demultipleksor s
pravilnostno tabelo



Kombinacijska vezja

Demultiplexorji

- več enakih demultiplexorjev s skupnimi izbirnimi vhodi lahko obravnavamo kot **vektorski demultiplexor**; navadni demultiplexor je v takšni terminologiji **skalarni demultiplexor**



$$\underline{Y} = \underline{m}^T \cdot \underline{d}$$

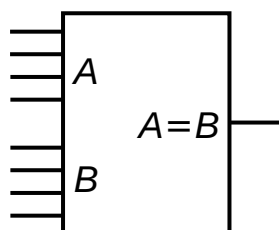
matrična enačba
vektorskega
demultiplexorja



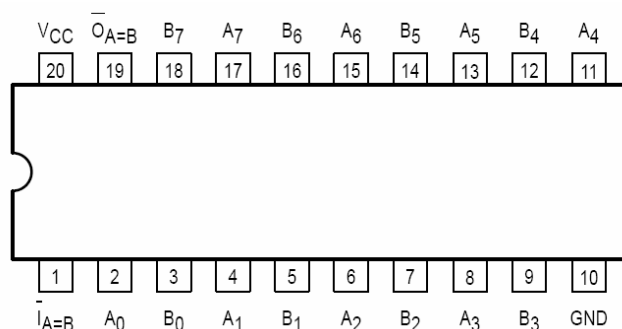
Kombinacijska vezja

Primerjalniki (komparatorji)

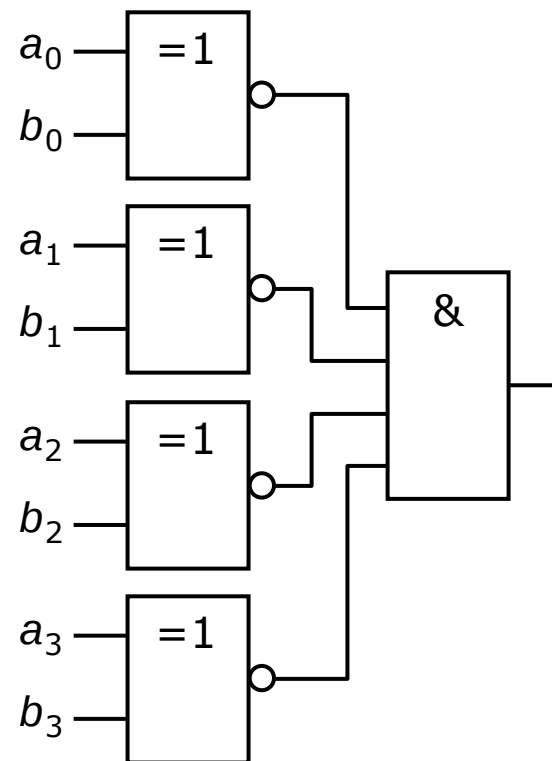
- n -bitni **primerjalnik enakosti** (*angl. identity comparator*) ima dva n -bitna (= vektorska) vhoda in en izhod, ki pove, ali sta števili na obeh vhodih enaki (t.j. ali so vsi istoležni biti obeh vhodov enaki):



4-bitni
primerjalnik
enakosti
(simbolna
shema)



8-bitni primerjalnik
enakosti MC74AC521
(Motorola)



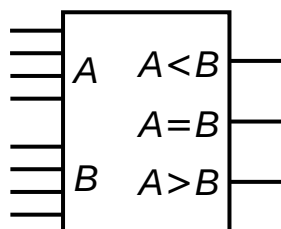
izvedba 4-bitnega
primerjalnika enakosti
z vrati NXOR in AND



Kombinacijska vezja

Primerjalniki (komparatorji)

- n -bitni **primerjalnik velikosti** (*angl. magnitude comparator*) ima dva n -bitna vhoda in tri izhode, ki odražajo razmerje velikosti števil na vloh:

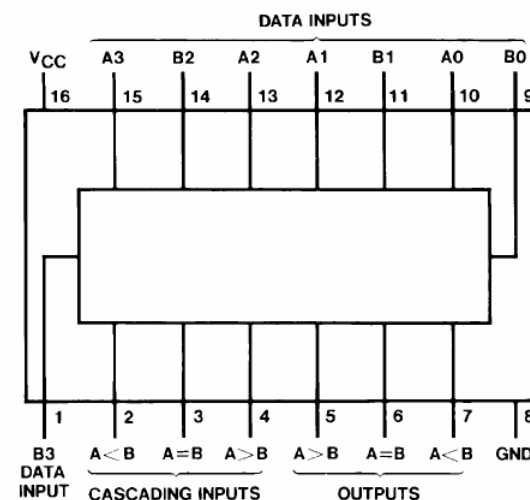


4-bitni
primerjalnik
velikosti
(simbolna
shema)

$$\begin{aligned}(A < B) = & \bar{a}_3 b_3 \\ & + \overline{(a_3 \oplus b_3)} \bar{a}_2 b_2 \\ & + \overline{(a_3 \oplus b_3)} \overline{(a_2 \oplus b_2)} \bar{a}_1 b_1 \\ & + \overline{(a_3 \oplus b_3)} \overline{(a_2 \oplus b_2)} \overline{(a_1 \oplus b_1)} \bar{a}_0 b_0\end{aligned}$$

$$(A = B) = \overline{(a_3 \oplus b_3)} \overline{(a_2 \oplus b_2)} \overline{(a_1 \oplus b_1)} \overline{(a_0 \oplus b_0)}$$

$$(A > B) = \overline{(A < B)} \overline{(A = B)}$$



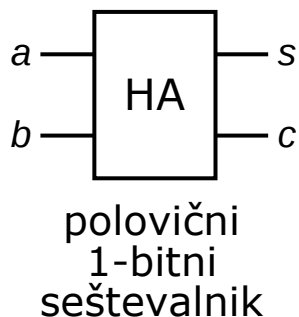
4-bitni primerjalnik
velikosti DM74LS85
(Natl. Semiconductor)



Kombinacijska vezja

Seštevalniki

- n -bitni seštevalnik (*angl. **adder***) ima dva n -bitna vhoda, na katera privedemo binarna zapisa dveh n -bitnih števil, na izhodu pa dobimo binarni zapis njune vsote in podatek, ali je pri seštetju prišlo do prenosa z najvišjega bita
- z 1-bitnim **polovičnim seštevalnikom** (*angl. **half adder, HA***) dobimo 1-bitno vsoto (s) in podatek o prenosu na višji bit (c)



a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$c = ab$$

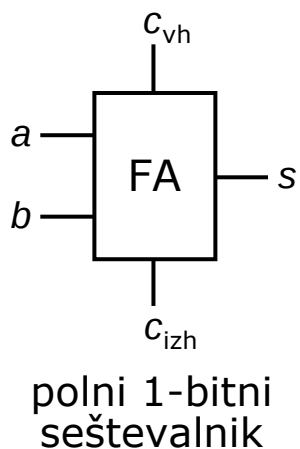
$$s = a \oplus b$$



Kombinacijska vezja

Seštevalniki

- 1-bitni **polni seštevalnik** (angl. **full adder, FA**) upošteva še morebitni prenos z nižjega bita (c_{vh})
- s kaskadno vezavo FA dobimo n -bitni **serijski (zaporedni) seštevalnik** (angl. **ripple adder**)



a	b	c_{vh}	c_{izh}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

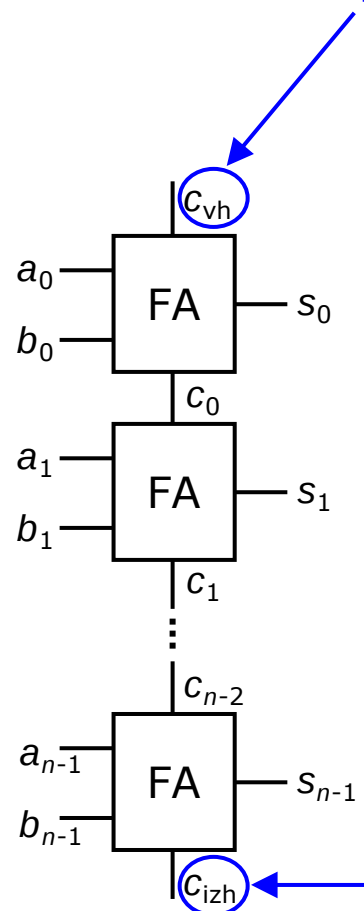
$$c_{izh} = ab + (a \oplus b)c_{vh}$$
$$s = a \oplus b \oplus c_{vh}$$

$$c_i = a_i b_i + (a_i \oplus b_i) c_{i-1}$$

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

enačbe in zgradba
 n -bitnega serijskega
seštevalnika

vhod za prenos pri prvem FA v kaskadi in
izhod za prenos pri zadnjem omogočata
združevanje takšnih seštevalnikov v večje





Kombinacijska vezja

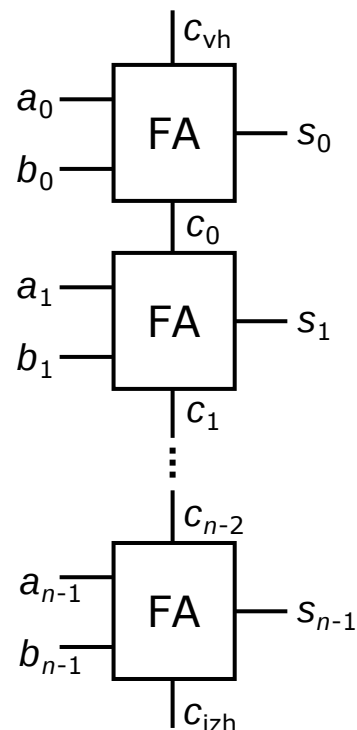
Seštevalniki

- v serijskem seštevalniku za izračun s_1 in c_1 potrebujemo s_0 in c_0 , za izračun s_2 in c_2 potrebujemo s_1 in c_1 , ...,
- ker ima odziv vsakega FA zakasnitev, se pravilne vrednosti $s_0, s_1, \dots, s_{n-1}$ ne pojavijo naenkrat, temveč druga za drugo
- celotni zakasnilni čas serijskega seštevalnika je tako odvisen od njegove velikosti (t.j. od n)

$$c_i = a_i b_i + (a_i \oplus b_i) c_{i-1}$$

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

enačbe in zgradba
 n -bitnega serijskega
seštevalnika





Kombinacijska vezja

Seštevalniki

- **paralelni (vzporedni) seštevalnik** (angl. **carry look-ahead adder**) vse prenose izračuna samo iz vhodnih podatkov:

označimo $c_{vh} = c_{-1}$, $G_i = a_i b_i$ in $P_i = a_i \oplus b_i$; tedaj je

$$c_0 = G_0 + c_{-1}P_0$$

$$c_1 = G_1 + c_0P_1 = G_1 + G_0P_1 + c_{-1}P_0P_1$$

$$c_2 = G_2 + c_1P_2 = G_2 + G_1P_2 + G_0P_1P_2 + c_{-1}P_0P_1P_2$$

$$\vdots$$

$$c_k = G_k + \sum_{j=0}^{k-1} \left(G_j \prod_{i=j+1}^k P_i \right) + c_{-1} \prod_{i=0}^k P_i$$

$$\vdots$$

$$s_0 = P_0 \oplus c_{-1}$$

$$s_1 = P_1 \oplus c_0$$

$$s_2 = P_2 \oplus c_1$$

$$\vdots$$

$$s_k = P_k \oplus c_{k-1}$$

$$\vdots$$



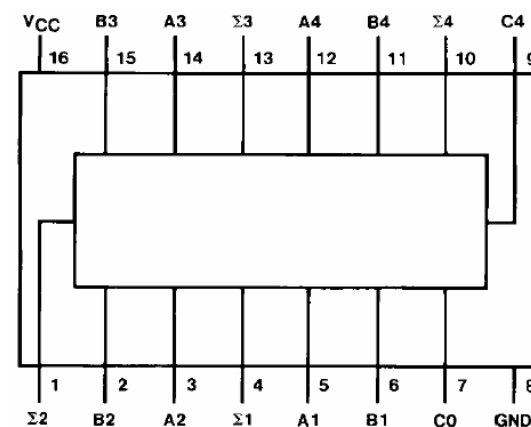
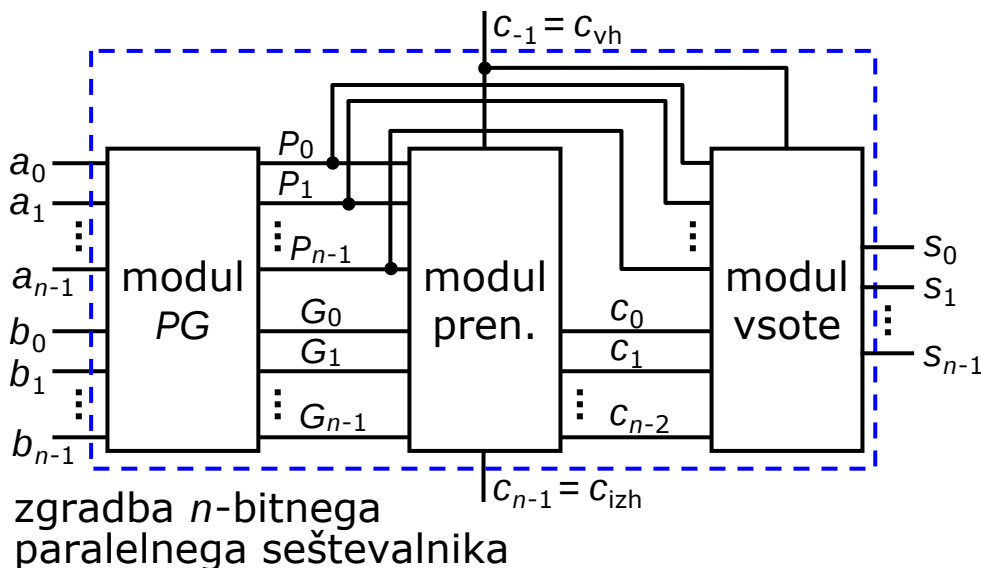
Kombinacijska vezja

Seštevalniki

- paralelni seštevalnik tako zgradimo iz treh delov:
 - modul PG** določi vse P_i in G_i
 - modul prenosov** določi vse c_i
 - modul vsote** določi vse s_i

$$c_k = G_k + \sum_{j=0}^{k-1} \left(G_j \prod_{i=j+1}^k P_i \right) + c_{-1} \prod_{i=0}^k P_i$$

$$s_k = P_k \oplus c_{k-1}$$



4-bitni paralelni seštevalnik
DM74LS283 (Fairchild)



Kombinacijska vezja

Seštevalniki

- za odštevanje uporabimo binarni zapis števil s predznakom: najpomembnejši bit predstavlja predznak (1-negativno, 0-pozitivno), število $-a$ pa je **dvojiški komplement** števila a , ki ga dobimo tako, da v binarnem zapisu števila a zamenjamo vse enice z ničlami in obratno, nato pa rezultatu prištejemo 1
- na ta način z n biti lahko zapišemo števila od $-2^{n-1}+1$ do $2^{n-1}-1$

-3	-2	-1	0	1	2	3
101	110	111	000	001	010	011

3-bitni zapis števil s predznakom

-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

4-bitni zapis števil s predznakom



Kombinacijska vezja

Seštevalniki

- razliko $(b - a)$ izračunamo tako, da številu b prištejemo dvojiški komplement a , morebitni prenos z najvišjega bita pa zanemarimo
- izračun razlik $(7 - 3)$ ter $(2 - 3)$ v 4-bitnem zapisu:

$$\begin{array}{r} 0111 \\ +1101 \\ \hline 10100 \end{array}$$

⇓

$$0100 \quad (= +4)$$

$$\begin{array}{r} 0010 \\ +1101 \\ \hline 01111 \end{array}$$

⇓

$$1111 \quad (= -1)$$

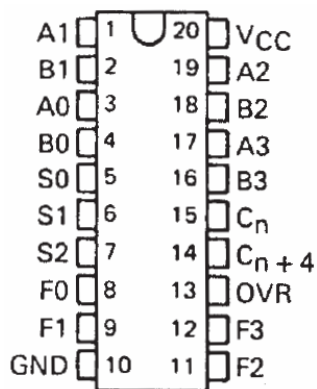
- če imata a in b enaka predznaka in sta znotraj območja izbranega n -bitnega zapisa, je opisani izračun razlike vselej pravilen
- v bolj splošnem primeru, ko imata a in b različna predznaka, pa lahko pride do preliva, t.j. dejanska vrednost $(b - a)$ preseže območje n -bitnega zapisa, dobljeni rezultat pa je zato napačen



Kombinacijska vezja

Aritmetično-logična enota (ALU)

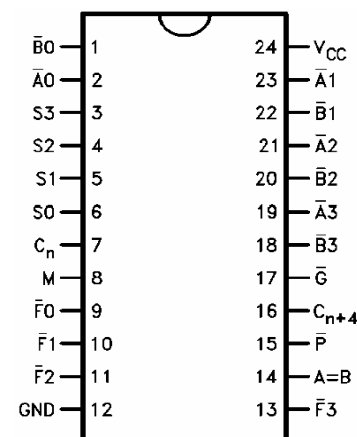
- aritmetično-logična enota (*angl. arithmetic logic unit, ALU*) je večnamensko vezje, ki izvaja logične in aritmetične operacije
- na podatkovne vhode dovedemo Booleove spremenljivke ali binarni zapis števil, z izbirnimi vhodi pa določimo operacijo, ki bo izvedena



4-bitna 8-operacijska
ALU SN74LS382-N
(Texas Instruments)

S_2	S_1	S_0	izhod
0	0	0	0
0	0	1	B minus A
0	1	0	A minus B
0	1	1	A plus B
1	0	0	$A \oplus B$
1	0	1	$A + B$
1	1	0	$A \cdot B$
1	1	1	1

seznam operacij
SN74LS382-N



4-bitna 32-operacijska
ALU 74LS181 (Signetics)