Kombinačné obvody

Pre súbor n logických premenných X_i, i = 1, 2, ... n existuje:

- **2^n rôznych vstupných kombinácií týchto premenných,**
- **2^{2n} rôznych logických funkcií Y_j**
  (pretože každá z nezávislých logických premenných môže nadobúdať dve hodnoty: (0, 1), teda j = 1, 2, ... 2^{2n}.

S rastúcim počtom logických premenných (n) teda počet navzájom rôznych logických funkcií, ktoré možno týmto premenným priradiť prudko narastá.

Je dokázané, že akúkoľvek logickú funkciu Y konečného počtu logických premenných je možné vyjadriť súborom vhodne vybraných logických funkcií Y_i jednej alebo dvoch logických premenných. Tento súbor logických funkcií nazývame úplný súbor logických funkcií.

Na každom úplnom súbore logických funkcií možno vybudovať logickú algebru.

Booleova algebra

Za funkcie zásadného významu sa z množstva funkcií považujú tzv. funkcie Y_1 až Y_{16}

A	B	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Z logických funkcií uvedených v týchto tabuľkách je možné vybrať niekoľko úplných súborov a vybudovať na nich logickú algebru. Najznámejší z úplných súborov, na ktorých bola vybudovaná logická algebra je súbor logických funkcií Y2, Y8, Y11, čiže logické funkcie AND, OR a NOT. Logická algebra vybudovaná na týchto funkciách sa podľa svojho autora Georgea Boolea nazýva Booleova alebo booleovská algebra.

Popis týchto funkcií je:

Symbol	Názov	Výrokový funktor	Booleov tvar
Y_1	konštanta falzum	nie je pravda	0
Y_2	konjunkcia (AND)	je A aj B	A.B
Y_3	priama inhibícia	je A a nie je B	$A.\overline{B}$
Y_4	asercia (opakovanie) A	je A	A
Y_5	spätná inhibícia	nie je A a je B	$\overline{A}.B$
Y_6	asercia (opakovanie) B	je B	B
Y_7	neekvivalencia (XOR)	je len A alebo len B	$A.\overline{B}+\overline{A}.B$
Y_8	disjunkcia (OR)	je A alebo B	A+B
Y_9	Piercova funkcia (NOR)	nie je A ani B	$\overline{A+B}$
Y_{10}	ekvivalencia	je A vtedy a len vtedy keď je B	$A.B+\overline{A}.\overline{B}$
Y_{11}	negácia (NOT) B	nie je pravda, že je B	$\overline{B}$
Y_{12}	spätná implikácia	ak je B, potom je A	$A+\overline{B}$
Y_{13}	negácia (NOT) A	nie je pravda, že je A	$\overline{A}$
Y_{14}	priama implikácia	ak je A, potom je B	$\overline{A}+B$
Y_{15}	Shefferova funkcia (NAND)	nie je A ani B	$\overline{A.B}$
Y_{16}	konštanta verum	je pravda	1
	XNOR		$\overline{\overline{A}.B+A.\overline{B}}$

Základné tvary booleovských funkcií

Booleovský súbor logických funkcií je úplný, teda vieme získať zápis ľubovoľnej logickej funkcie pomocou booleovských funkcií a to v zásade v dvoch tvaroch:

štandardný súčtový tvar,
štandardný súčinový tvar.

Štandardný súčtový tvar získame tak, že v každom riadku pravdivostnej tabuľky, kde má závislá premenná pravdivostnú hodnotu 1 vynásobíme priamu alebo negovanú pravdivostnú hodnotu každej nezávislej premennej X_i podľa toho, či má táto premenná pravdivostnú hodnotu 1 alebo 0 prislúchajúcou premennou, získané súčiny vynásobíme, takto získané súčiny sčítame.
(=normálna disjunktívna (súčtová) forma – NDF)

Štandardný súčinový tvar získame tak, že v každom riadku pravdivostnej tabuľky, kde má závislá premenná pravdivostnú hodnotu **0** vynásobíme priamu alebo negovanú pravdivostnú hodnotu každej nezávislej premennej X_i podľa toho, či má táto premenná pravdivostnú hodnotu 1 alebo 0 prislúchajúcou premennou, získané súčiny sčítame, takto získané súčty vynásobíme.

Vo všeobecnosti možno povedať, že ak nezávislá premenná nadobúda prevažne pravdivostné hodnoty 0, použijeme súčtovú normálovú formu, ak nadobúda prevažne pravdivostné hodnoty 1, použijeme súčinovú normálovú formu

(= normálna konjunktívna (súčinová) forma – NKF)

PRÍKLAD

Majme funkciu Y (A ,B,C)

ktorej pravdivostná tabuľka je:

A	B	C	Y
1	1	1	1
1	0	1	0
1	1	0	0
1	0	0	1
0	1	1	1
0	0	1	1
0	1	0	1
0	0	0	1

Napíšte

1. súčtovú normálovú formu,
2. súčinovú normálovú formu tejto funkcie.

1. Súčtová normálová forma tejto funkcie je

$$Y(A,B,C) = A.B.C + A.\overline{B}.\overline{C} + \overline{A}.B.C + \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.\overline{B}.\overline{C}$$

2. Súčinová normálová forma tejto funkcie je

$$Y(A,B,C) = (\overline{A} + B + \overline{C}).(\overline{A} + \overline{B} + C)$$

Ak sa v každom terme normálnej formy vyskytujú všetky písmená premenných (tu – A aj B aj C) danej funkcie (tu Y) jedná sa o tzv. úplnú normálnu formu. (UNDF alebo UNKF)

Pravidlá a zákony zjednodušovania booleovských funkcií

Pri určovaní pravdivostnej logickej funkcie je dôležité, aby bola táto funkcia čo najjednoduchšia, preto po procese jej zostavenia nasleduje proces jej zjednodušenia.

Podobne ako v normálnej algebre, aj v booleovskej algebre je možné upravovať a zjednodušovať logické funkcie podľa určitých pravidiel a zákonov.

Pravidlá zjednodušovania logických funkcií:

- najprv upravujeme logické výrazy vo vnútri zátvoriek,
- potom upravujeme logickú funkciu podľa priority logických operátorov, pričom priority jednotlivých logických operátorov sú:
  - 1. negácia,
  - 2. logický súčin,
  - 3. logický súčet.

$A + B = B + A$ $A \cdot B = B \cdot A$

$A + (B + C) = (A + B) + C$ $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

$A \cdot (B + C) = A \cdot B + A \cdot C$

$\overline{a + b} = \overline{a} \cdot \overline{b}$

$\overline{a \cdot b} = \overline{a} + \overline{b}$

$a + 1 = 1$

$a \cdot 1 = a$

$a + 0 = a$

$a \cdot 0 = 0$

$a \cdot a = a$

$a + \overline{a} = 1$

Formalizovanie logických výrokov

- Základné identity
- Komutatívny zákon (zámena sčítancov)
- Asociatívny (o združovaní)
- Distributívny (o roznásobení)
- De Morganove identity

Zákony zjednodušovania logických funkcií:

Zákon	Súčtový tvar	Súčinový tvar
komutatívny	$A+B=B+A$	$A.B=B.A$
asociatívny	$A+(B+C)=(A+B)+C$	$A.(B.C)=(A.B).C$
distributívny	$A+(B.C)=(A+B).(A+C)$	$A.(B+C)=(A.B)+(A.C)$

De Morganov	$\overline{A+B}=\overline{A}.\overline{B}$	$\overline{A.B}=\overline{A}+\overline{B}$
absorbcie	$A+(A.B)=A$	$A.(A+B)=A$
absorbcie negácie	$A+\overline{A}.B=A+B$	$A.\left(\overline{A}+B\right)=A.B$
vylúčenia	$A+\overline{A}=1$	$A.\overline{A}=0$
idempotencie	$A+A=A$	$A.A=A$
neutrálnosti 0 a 1	$A+0=A$	$A.1=A$
agresívnosti 0 a 1	$A+1=1$	$A.0=0$
dvojitej negácie A	$\neg\neg A=A$	$\neg\neg A=A$

Kombinačné obvody

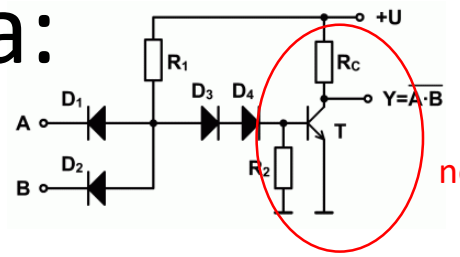
Jednoduché kombinačné obvody (hradlá) slúžia na realizáciu základných logických operácií a základnými sú:

hradlo NOT,
hradlo AND, NAND, hradlo OR, NOR,
hradlo XOR, XNOR,
hradlo AND OR INVERT.

Zložitejšie kombinačné obvody (aritmetické jednotky) slúžia na realizáciu zložitejších aritmetických logických operácií a základnými sú:

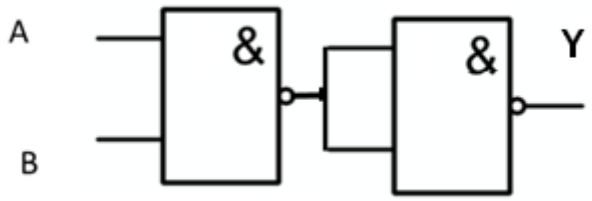
sčítačka, násobička,
multiplexor, demultiplexor,
prepínač, komparátor, kóder, dekóder, generátor parity,
aritmeticko – logická jednotka.

Zjednodušenia:



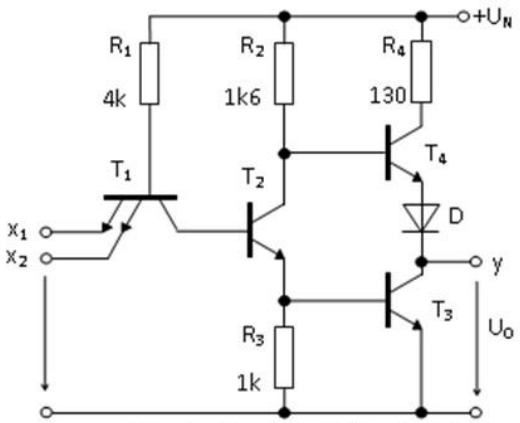
negátor

(DTL)



a	b	AND	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NAND (TTL)

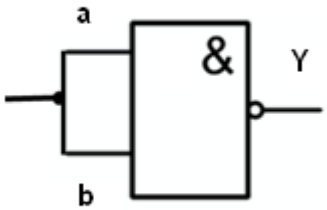


Zapojenie základného logického obvodu NAND s dvomi vstupmi

AND

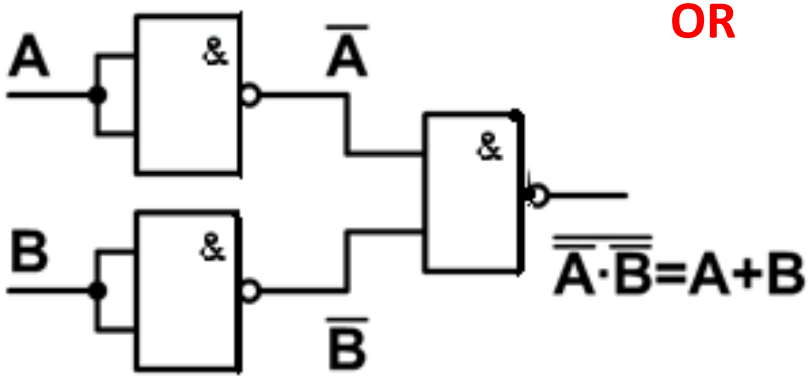
A	B	$\overline{A+B}$	Y= $\overline{A+B}$
0	0	**1**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

NOT



NOT

OR

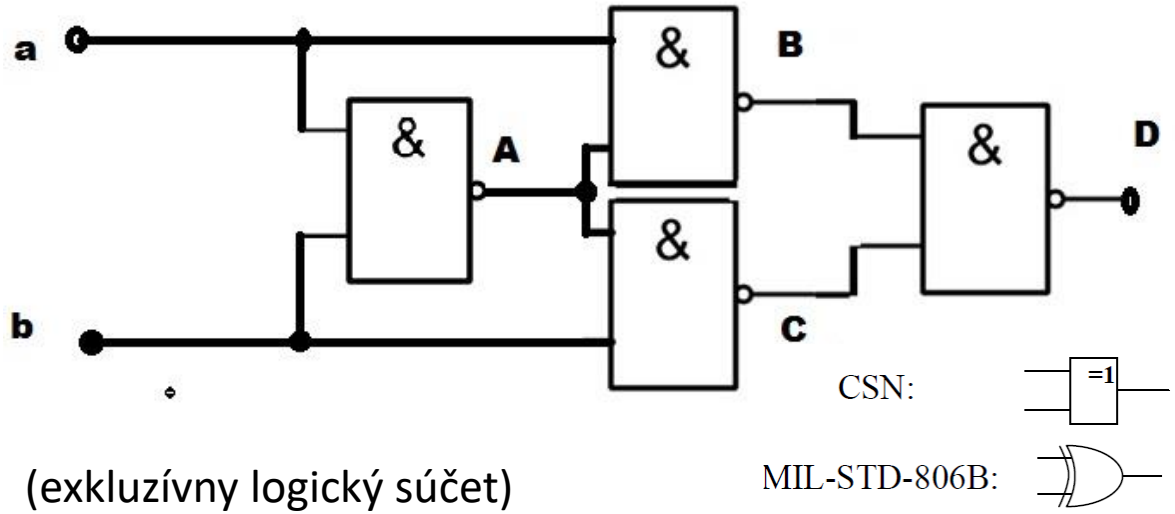


A	B	$\overline{A}$	$\overline{B}$	$\overline{(\overline{A}.\overline{B})}$	Y = A + B
0	0	1	1	**0**	**0**
0	1	1	0	**1**	**1**
1	0	0	1	**1**	**1**
1	1	0	0	**1**	**1**

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

S využitím de Morganovej identity...

XOR sa dá realizovať pomocou NAND nasledovne:



(exkluzívny logický súčet)

$A \oplus B$

Úpravou dostaneme to isté:

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

$$B = \overline{(a).(\overline{a.b})} \equiv \bar{a} + a.b$$

$$C = \overline{(b).(\overline{a.b})} \equiv \bar{b} + a.b$$

$$D = \overline{[\bar{a} + a.b].[\bar{b} + a.b]}$$

$$\equiv \overline{[\bar{a} + a.b]} + \overline{[\bar{b} + a.b]}$$

$$\equiv a.\overline{a.b} + b.\overline{a.b}$$

a	b	A	B	C	D
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

$\overline{a.b} = 1$

$\overline{a.b} = 0$

0 + 1 $\overline{a.b} = 1$

1 + 0

$$\equiv a.(\bar{a} + \bar{b}) + b.(\bar{a} + \bar{b})$$

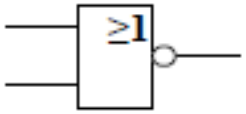
$$\equiv a\,.\,\bar{a} + a.\bar{b} + b.\bar{a} + b.\bar{b}$$

0 *teda* *0*

$$\boldsymbol{D \equiv a.\bar{b} + b.\bar{a}}$$

(vidíme, že ak $a \neq b$ XOR =1 – „hovoríme tomu „ neekvivalencia)

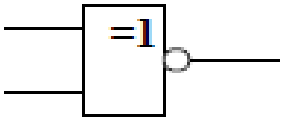
Hradlo NOR (Peirceova funkcia)



A	*B*	*A* NOR *B*
0	0	1
0	1	0
1	0	0
1	1	0

$$\overline{A + B}$$

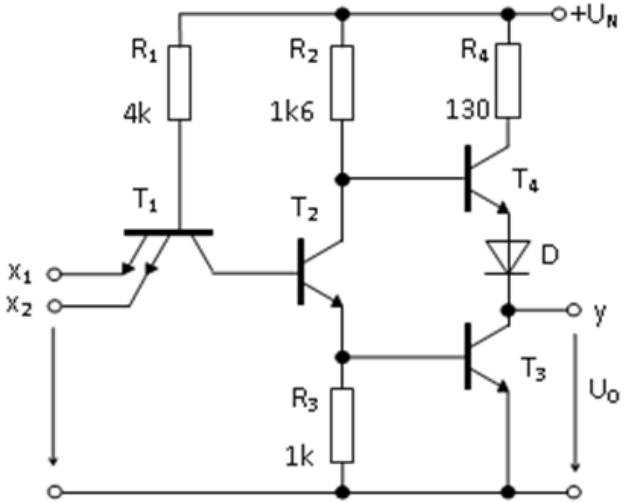
Hradlo XNOR



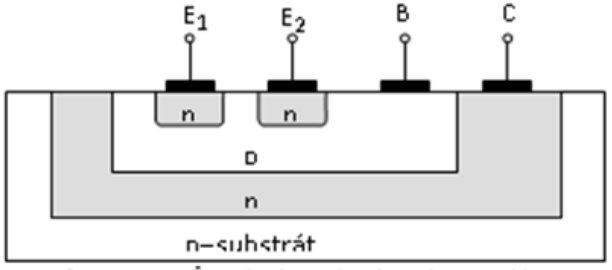
A	*B*	*A* XNOR *B*
0	0	1
0	1	0
1	0	0
1	1	1

$$\overline{A \oplus B}$$

Integrované obvody



Zapojenie základného logického obvodu NAND s dvomi vstupmi



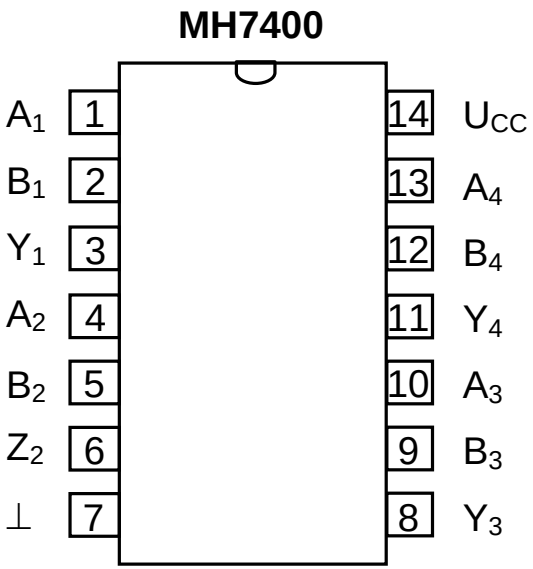
Štruktúra dvojemitorového tranzistora

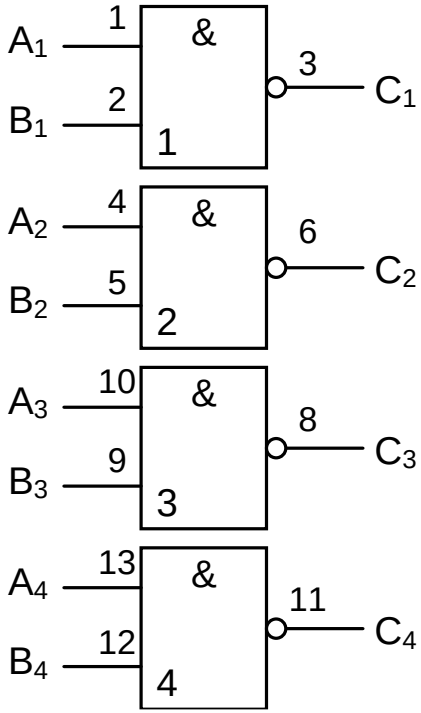
7400

Keď na vstupy x_1 a x_2 je privedená úroveň H, tranzistory T_2 a T_3 sú v saturácii a T_4 je uzavretý. Ak sa niektorý vstup x1, x2 zmení na úroveň L, T_2 a T_3 sa uzavrú, napätie na bázi T_4 sa zväčšuje a mení sa aj výstupná úroveň U_O a to dovtedy, pokiaľ nedosiahne hodnotu napájacieho napätia zmenšenú o hodnotu úbytkov na dióde D, tranzistore T_4 a rezistore R_4. Hodnota výstupného napätia je asi 3,5V (úroveň H). Počas zmeny vstupného napätia sa nabíja zaťažovacia kapacita cez rezistor R_4, otvorený tranzistor T_4 a diódu D.

MH7400

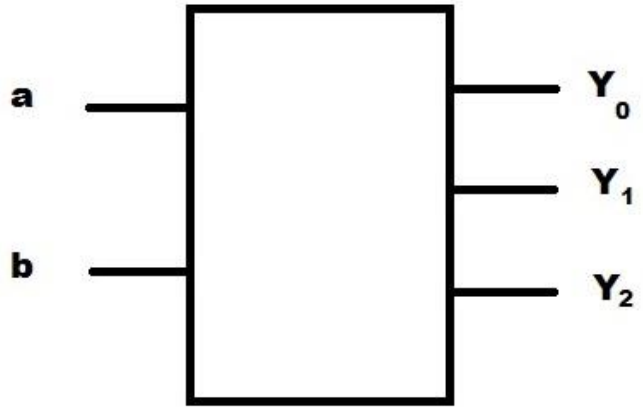
- ***Schematická značka obvodu a zapojenie vývodov***





Jednobitový komparátor (cvičenie)

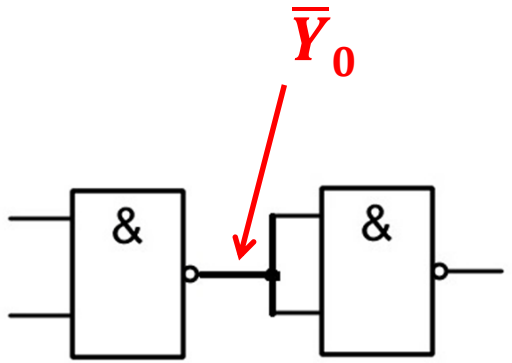
Dovoľuje rozhodnúť o vzájomnom vzťahu dvoch jednobitových registrov a, b tj. odpovedá na otázku:

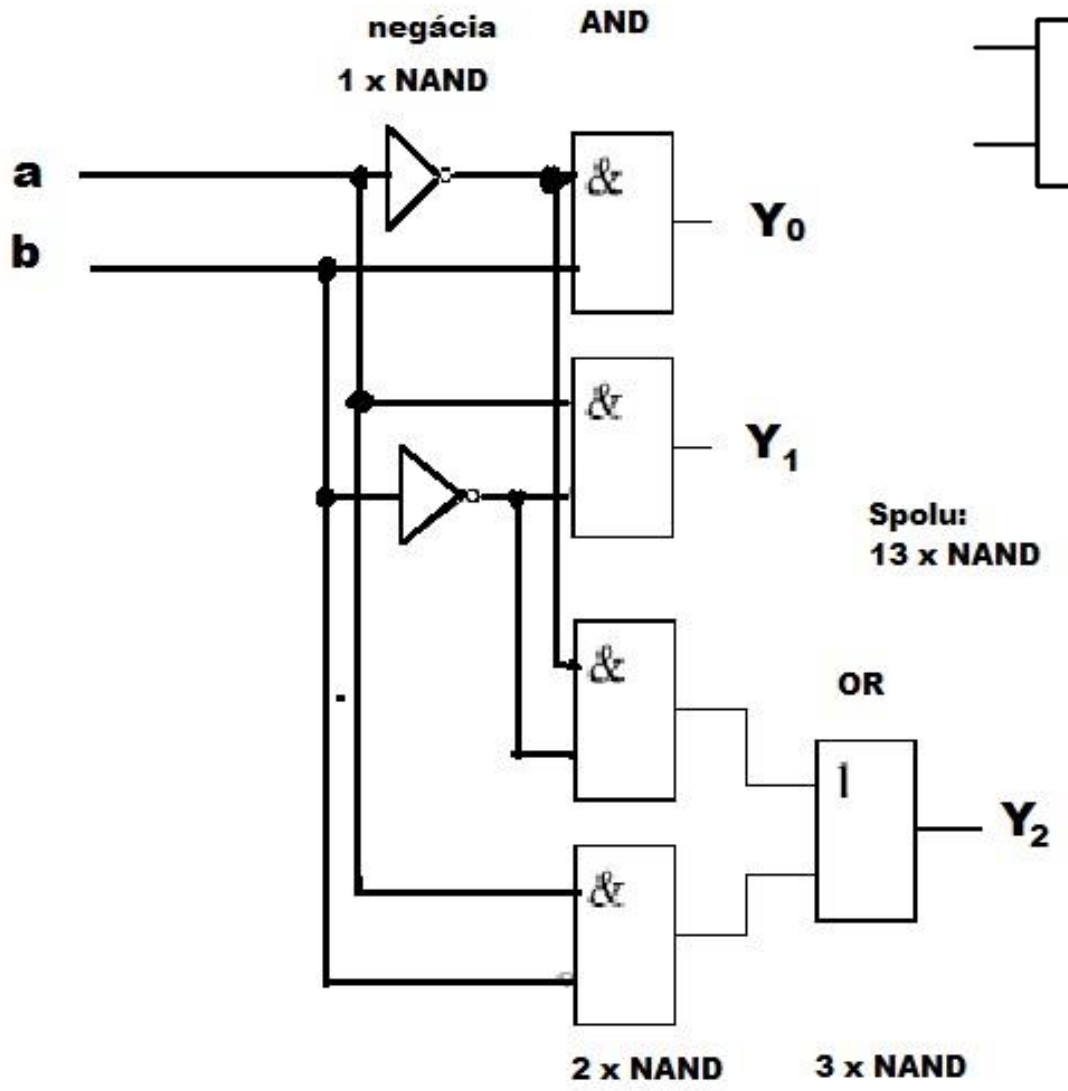


a < b
a > b ?
a = b

a	b	$\overline{a}$	$\overline{b}$	$\overline{a}.b$ Y_0	$a.\overline{b}$ Y_1	$\overline{a}.\overline{b}$	$a.b$	$\overline{a}.\overline{b} + a.b$ Y_2
0	0	1	1	0	0	1	0	1
0	1	1	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0
1	1	0	0	0	0	0	1	1

Logika zapojenia





podobne $\overline{Y}_1$

Kedže jednou z identít Booleovej algebry je: $a + \overline{a} = 1$

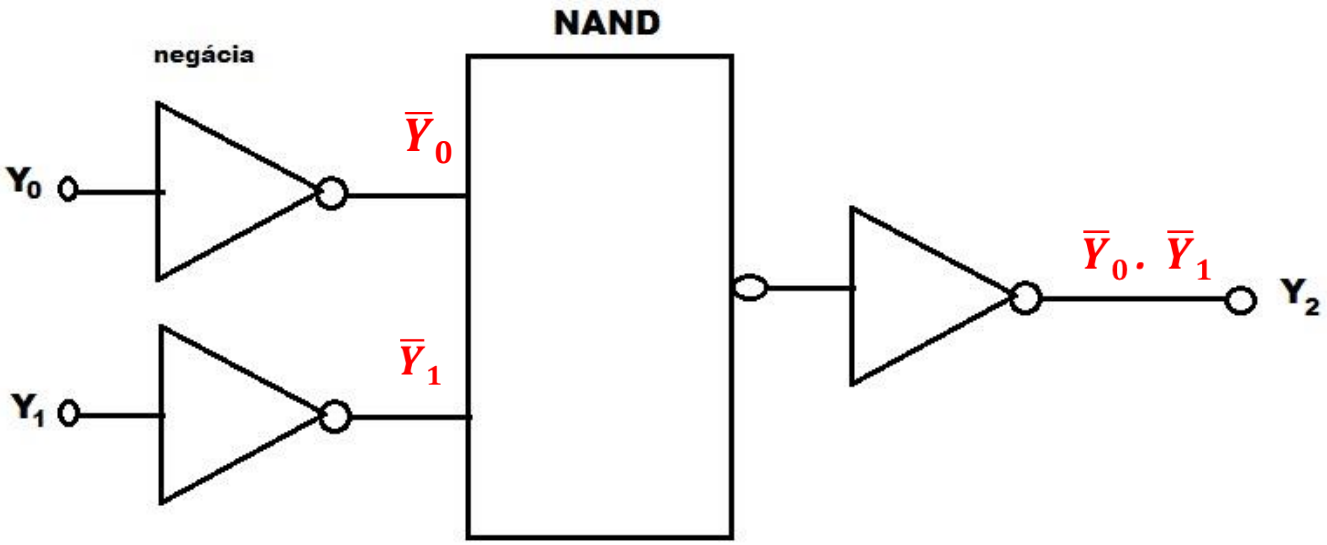
pre výstupné hodnoty nášho komparátora platí: $Y_0 + Y_1 + Y_2 = 1$

(tj. jednotkový je vždy jeden z výstupov podľa okamžitého stavu vstupov)

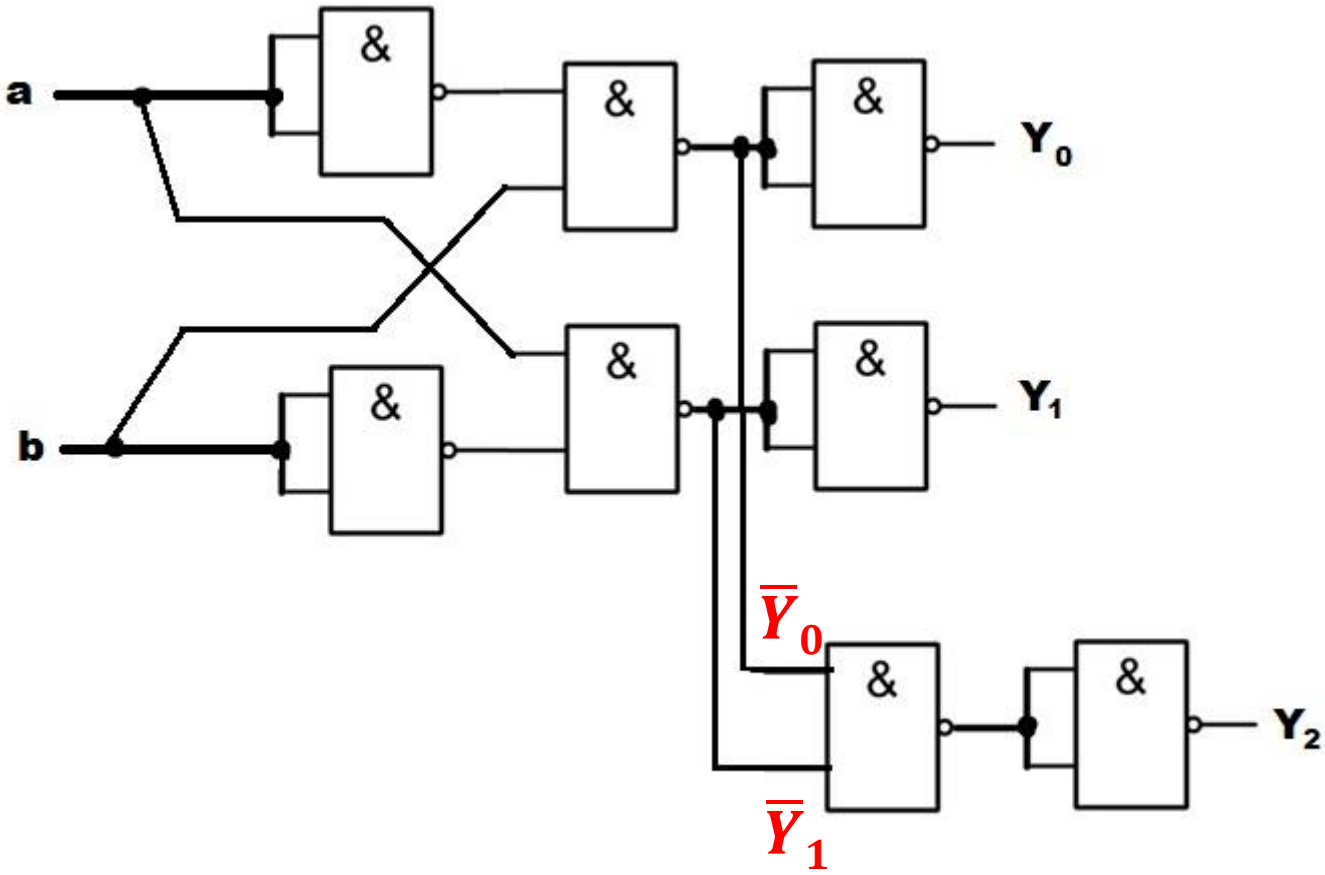
druhú rovnicu môžeme prepísať aj ako $(Y_0 + Y_1) + \overline{(Y_0 + Y_1)} = 1$

použitím De Morganovho pravidla môžeme písať: $Y_2 = \overline{Y}_0 . \overline{Y}_1$

potom zapojenie pre prípad keď $a = b$ môže vyzerať nasledovne:



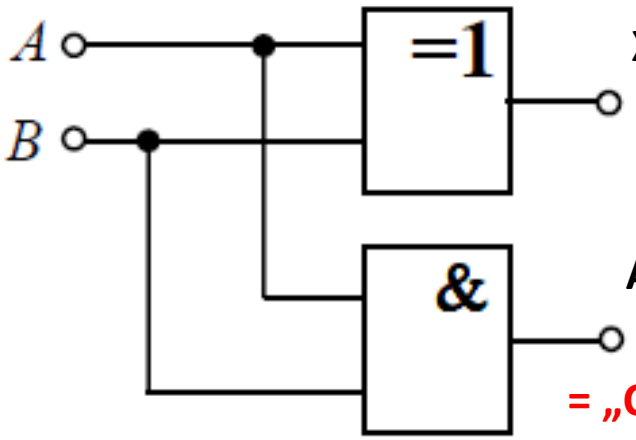
Zjednodušené riešenie



Jednobitová polovičná sčítačka

Realizácia:

alebo



XOR

Y

= jednobitový aritmetický súčet

AND

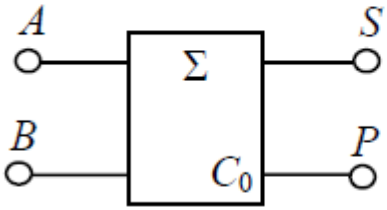
C

= „Curry flag“! príznak prenosu

Pravdivostná tabuľka:

VSTUP		VÝSTUP	
B	***A***	***S***	***P***
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Schematická značka



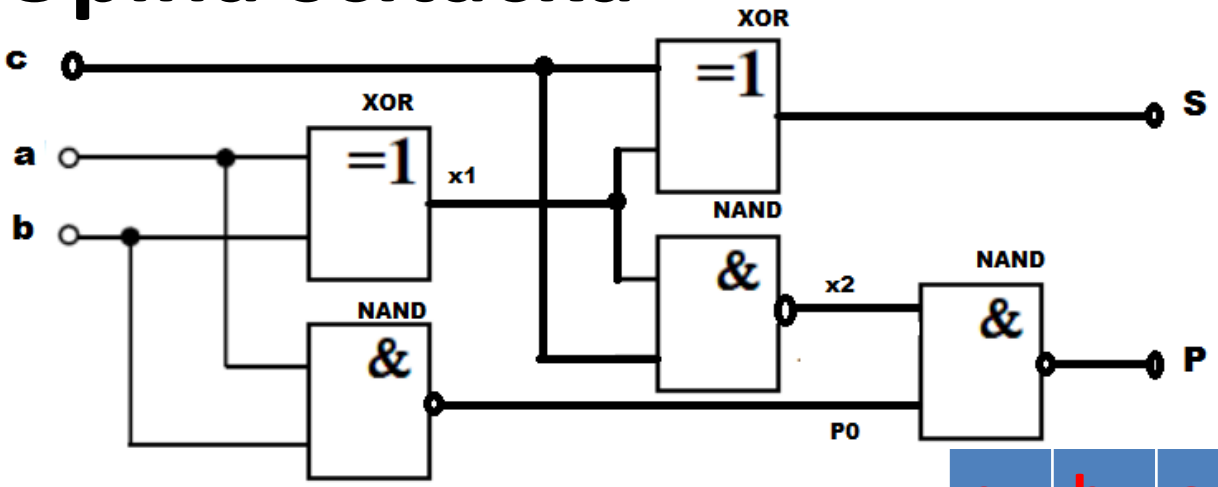
Booleovský zápis:

$$S = \overline{A} \,.\, B + A \,.\, \overline{B} \qquad C = A \,.\, B$$

C=P=príznak prenosu

odovzdáva síce príznak prenosu do vyššieho rádu, nie je však schopná prijať ho z nižšieho rádu

Úplná sčítačka



Umožňuje sčítanie dvoch jednobitových binárnych čísel s pripočítaním prenosu z predchádzajúceho rádu.

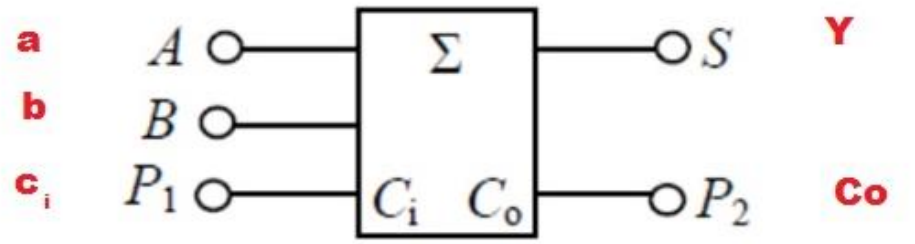
Vstupom sú tri jedno-bitové sčítance **A, B, C_{in}** (Carry-in) (resp. **P1)**,

výstupom sú 1- bitový súčet **S** a 1-bitový príznak prenosu do vyššieho rádu **C_{out}** (Carry-out) aj (**P2**).

c	b	a	x1 S0	x2	P0	S	P
0	0	0	0	1	1	0	0
0	0	1	1	1	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	0	1
1	0	0	0	1	1	1	0
1	0	1	1	0	1	0	1
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	1

c + b + a = Súčet + Prenos

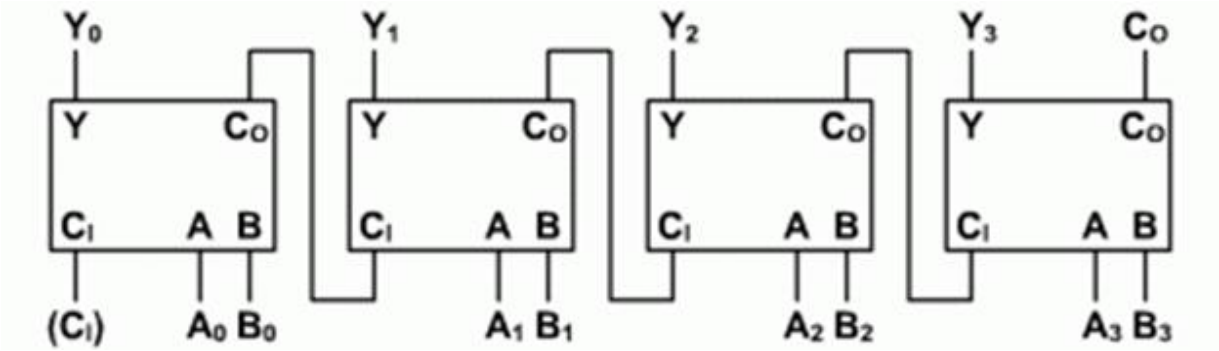
Schématická značka:



Boolovský zápis : vid. „Poznámky k prednáške (L3)“

... Detailné vysvetlenie si môžete naštudovať z prednášky L4!

Zapojenie štvorbitovej sčítačky



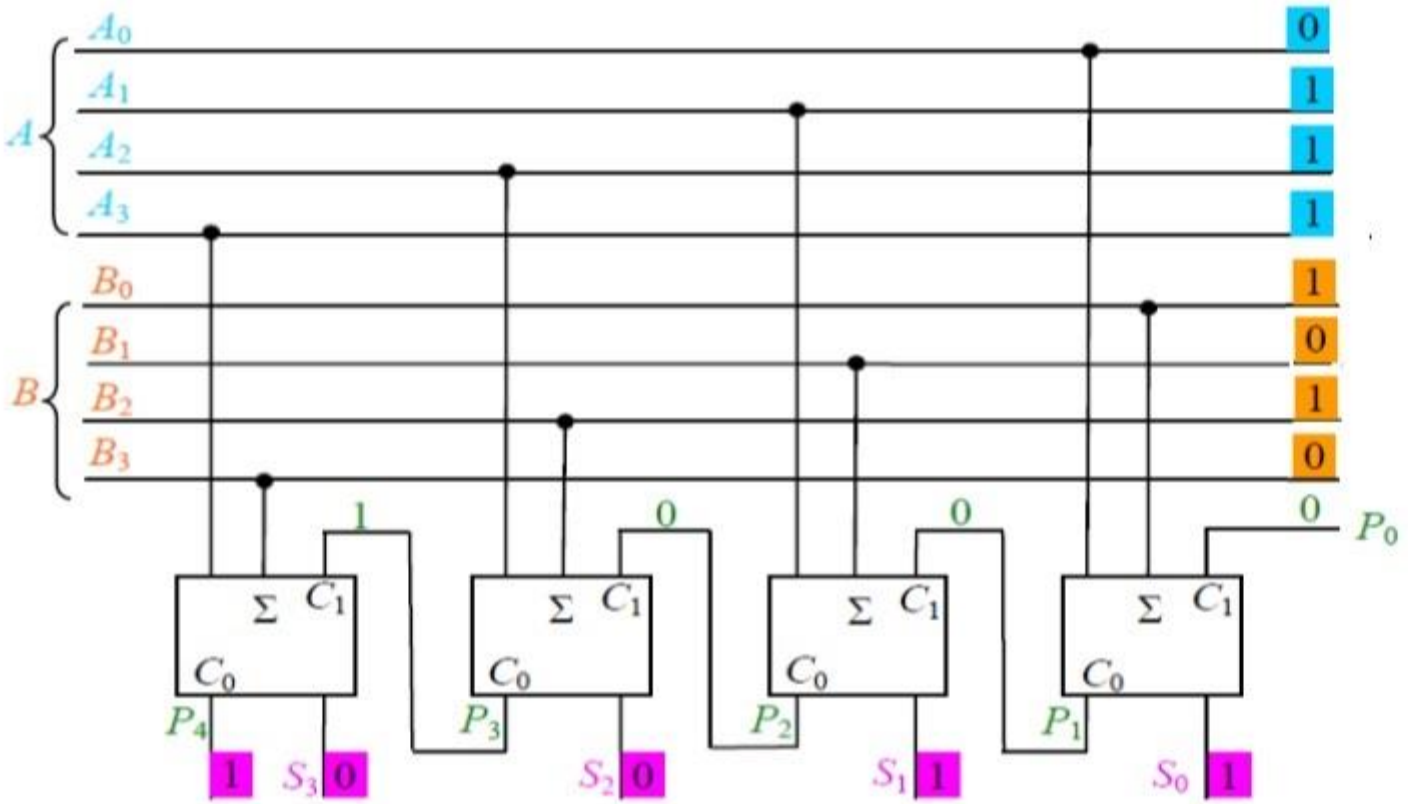
Viacbitové sčítačky

N-bitová sčítačka s propagáciou prenosu resp. so sériovým prenosom (Ripple Carry Adder – RCA) je zostavená z N úplných 1-bitových sčítačiek ich jednoduchým zreťazením. Vstupom tejto sčítačky sú potom dve N-bitové čísla + 1-bitový prenos z predchádzajúceho rádu, výstupom sú N-bitový súčet vstup- ných N-bitových čísel + 1-bitový prenos z predchádzajúceho rádu (spolu N+1bitov).

Poznámka: Pretože každá sčítačka musí „čakať“ na príznak prenosu od sčítačky predchádzajúceho bitu a toto oneskorenie výrazne obmedzuje jej priepustnosť t.j. maximálny možný počet sčítaní za jednotku času!!

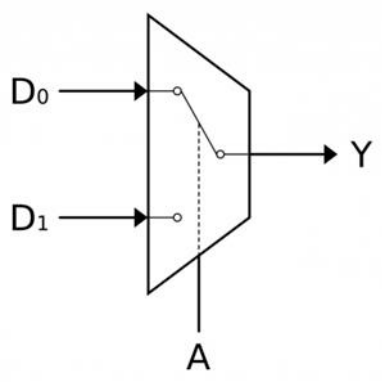
Pre 4-bitovú paralelnú sčítačku so sériovým prenosom je

sčítanec: $A =$	1	1	1	0	
sčítanec: $B =$	0	1	0	1	
	1	1	0		← prenos
	P_4	P_3	P_2	P_1	P_0
súčet	1	0	0	1	1



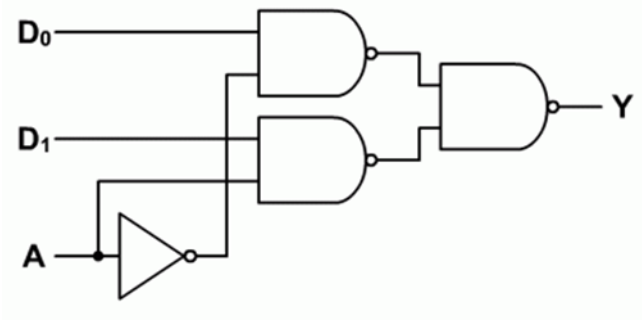
Multiplexor a demultiplexor

Funkcia multiplexoru ako prepínača



A	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

3 x NAND

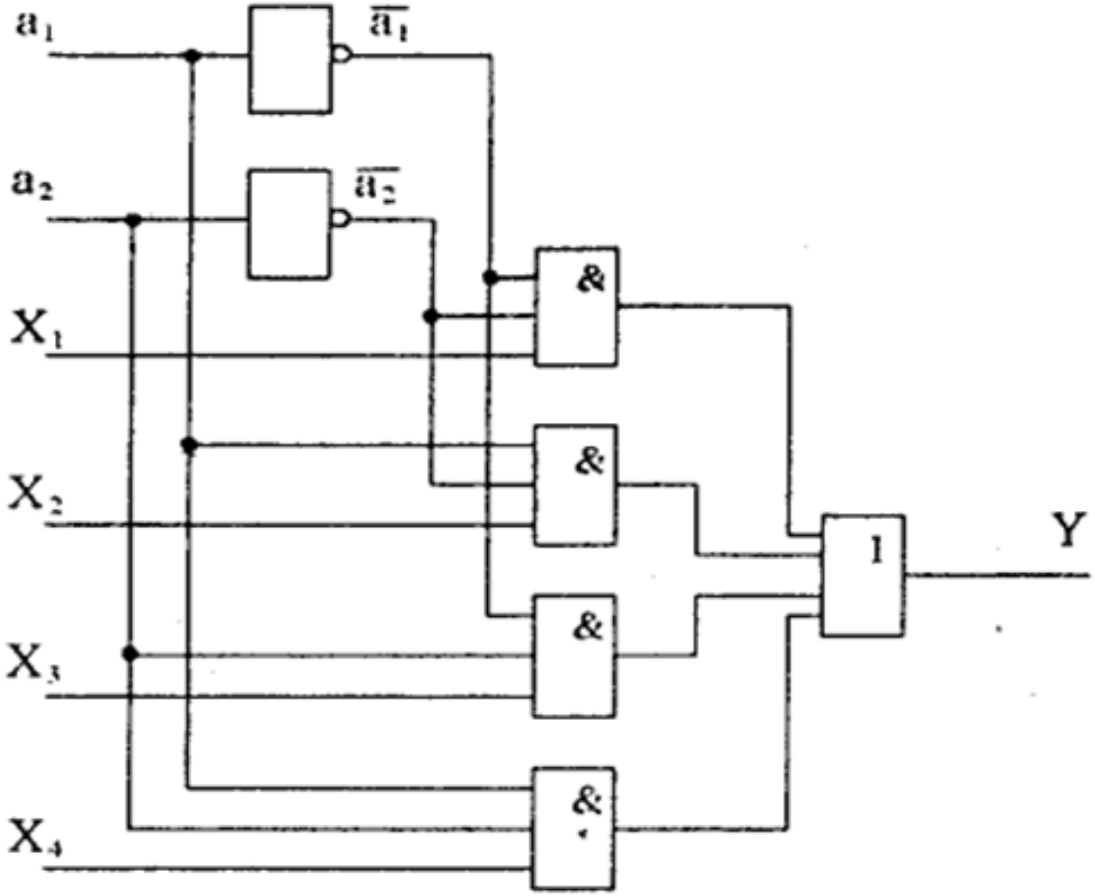


Ak A = 0 výstup Y = D_0 a ak A = 1 výstup Y = D_1 !

Teda podľa hodnoty riadiaceho signálu A je na výstup Y pripojený buď signál D_0 alebo D_1. Takáto funkcia sa dá popísať nasledovnou tabuľkou:

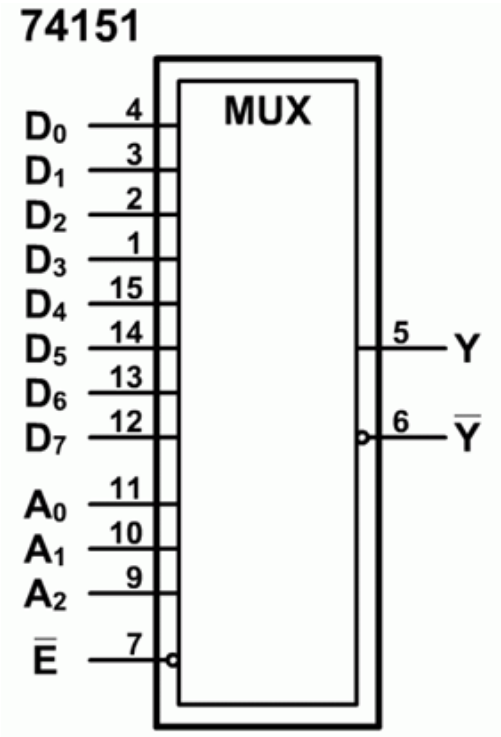
A	Y
0	D_0
1	D_1

Prepínač – štyri vstupy jeden výstup



A1	A2	Y
0	0	X1
0	1	X2
1	0	X3
1	1	X4

74xx151 (8 -vstupový multiplexor)

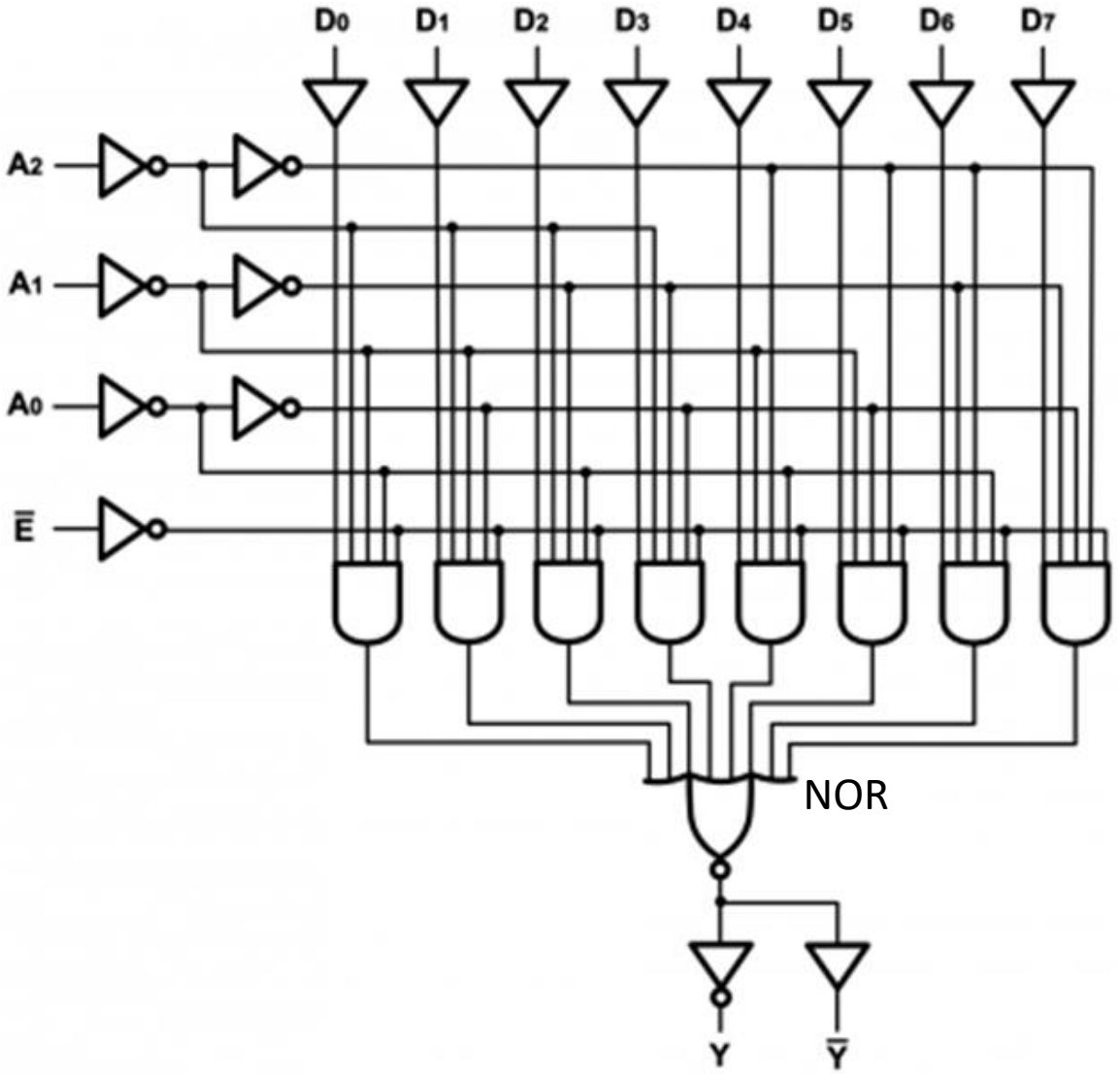


+ Ucc (16)
+ GND (8)

vstup E uvoľňuje/blokuje (0/1) obvod

Vstupy												Výstupy	
E	A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y	Y
H	X	X	X	X	X	X	X	X	X	X	X	L	H
L	L	L	L	L	X	X	X	X	X	X	X	L	H
L	L	L	L	H	X	X	X	X	X	X	X	H	L
L	L	L	H	X	L	X	X	X	X	X	X	L	H
L	L	L	H	X	H	X	X	X	X	X	X	H	L
L	L	H	L	X	X	L	X	X	X	X	X	L	H
L	L	H	L	X	X	H	X	X	X	X	X	H	L
L	L	H	H	X	X	X	L	X	X	X	X	L	H
L	L	H	H	X	X	X	H	X	X	X	X	H	L
L	H	L	L	X	X	X	X	L	X	X	X	L	H
L	H	L	L	X	X	X	X	H	X	X	X	H	L
L	H	L	H	X	X	X	X	X	L	X	X	L	H
L	H	L	H	X	X	X	X	X	H	X	X	H	L
L	H	H	L	X	X	X	X	X	X	L	X	L	H
L	H	H	L	X	X	X	X	X	X	H	X	H	L
L	H	H	H	X	X	X	X	X	X	X	L	L	H
L	H	H	H	X	X	X	X	X	X	X	H	H	L

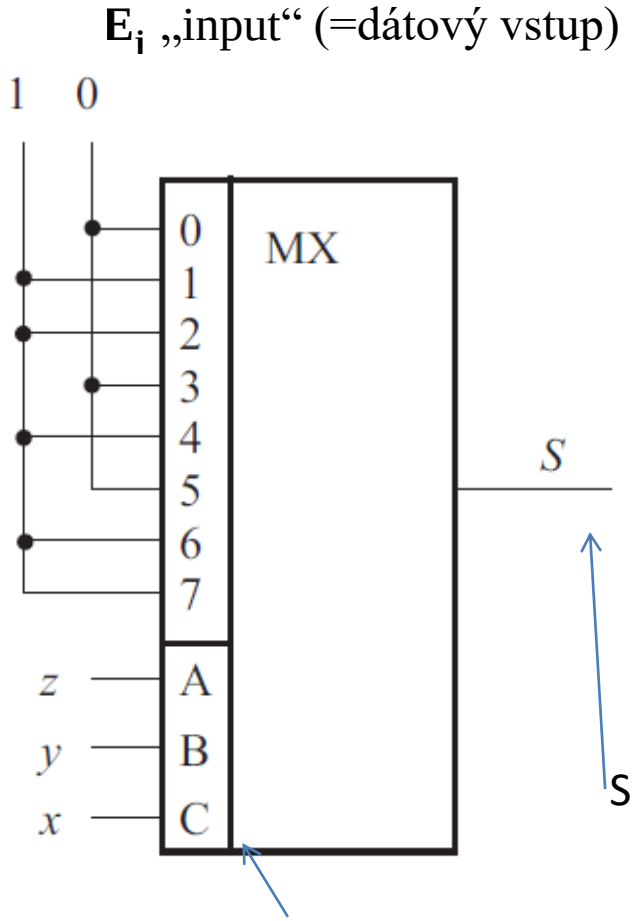
Funkčná schéma 74151



8 x AND

Generovanie logických funkcií (pomocou multiplexora)

x	y	z	f_j
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Stačí túto funkciu považovať za funkciu premenných

(C = x, B = y, A = z)

a pre $\mathbf{j = j_1.2^2 + j_2.2^1 + j_3.2^0}$ na informačný vstup $\mathbf{E_j}$ pripojiť konštantu **0** práve vtedy, keď

$\mathbf{f\,(j_1, j_2, j_3) = 0,}$

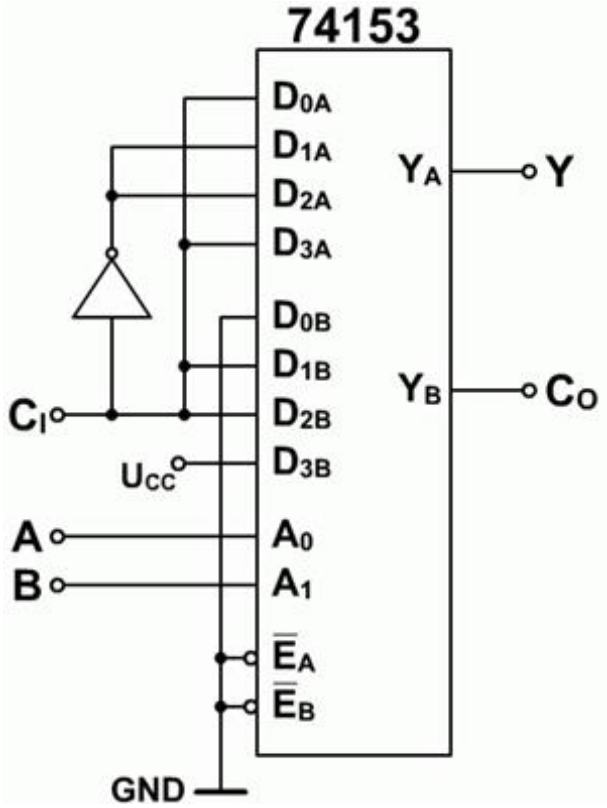
a konštantu **1** práve vtedy, keď

$\mathbf{f\,(j_1, j_2, j_3) = 1.}$

Úplná jednobitová sčítačka (realizovaná pomocou multiplexora)

C_I	B	A	C_O	Y
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

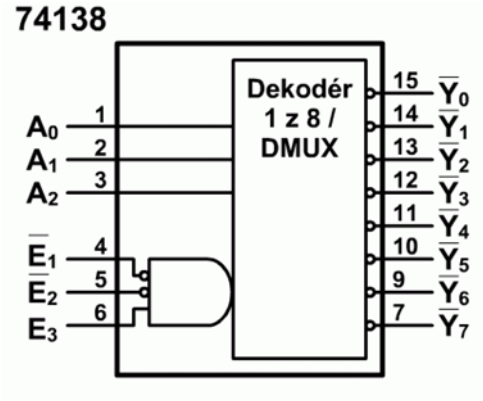
Logické funkcie Y a C_O realizujeme pomocou dvoch štvorvstupových multiplexorov v integrovanom obvode 74153 a jedného invertora



Demultiplexor

Zatial čo multiplexor je elektronický prepínač vstupných signálov na jeden výstup, **demultiplexor prepína jeden vstupný signál na niekoľko výstupov**. Príslušný výstup sa taktiež vyberá adresovým signálom. Čiže ak prenášame signál multiplexovaný z n-kanálov po jednom (dátovom) „ drôte“ (optický kábel, telefónna linka, satelitné spojenie), môžeme ho späť transformovať do n-drôtov pomocou demultiplexoru. K tomu je však potrebná aj znalosť (prenos) adresových signálov.

Reálny príklad



Demultiplexor môžeme použiť aj na riadenie displejov (LED, LCD). V takomto prípade, v každom okamžiku musí byť aktívna práve jedna číslica (znak resp. riadok displeja) v závislosti na privedenej adrese.
Zmena adresy ale musí byť dostatočne rýchla, aby to prepínanie nebolo postrehnuteľné ľudským okom.

Mux_DMux [35]

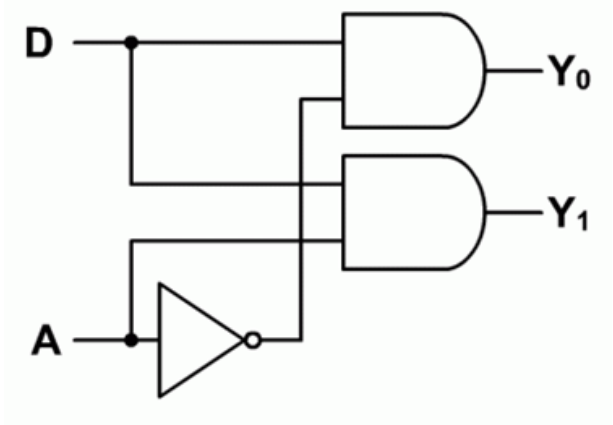
Pravdivostná tabuľka dvojkanálového demultiplexoru.
Pre A = 0 je logický signál privádzaný na výstup Y_0 a pre A = 1 na výstup Y_1

A	D	Y_0	Y_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

Čiže:

$$Y_0 = \bar{A}D$$
$$Y_1 = AD$$

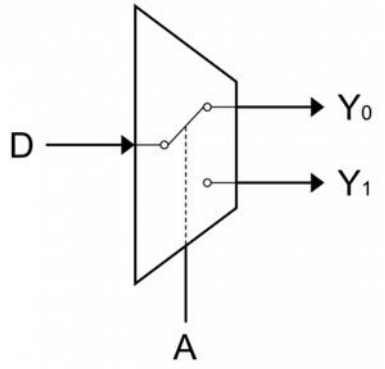
ZAPOJENIE: AND



Skrátený zápis :

A	Y_0	Y_1
0	D	0
1	0	D

Schématická značka:



Rozdiel medzi demultiplexorom a dekodérom

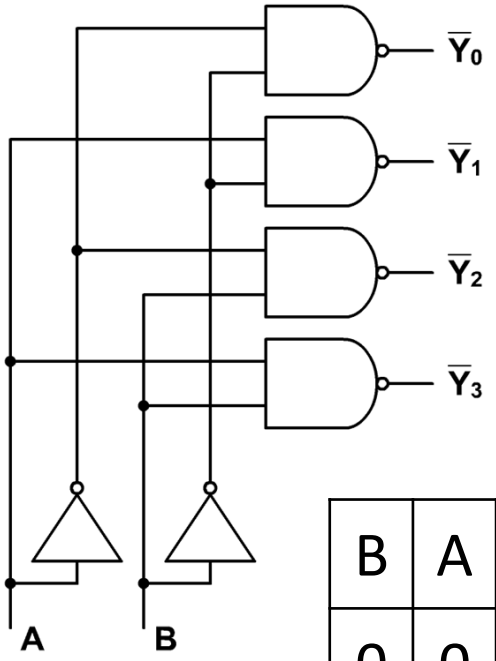
V zásade aj predchádzajúce zapojenie, môže plniť funkciu dekodéru, ktorý by prevádzal jednobitové binárne číslo zo vstupu **A** na kód **1 z 2**.

Vstup **D** v tomto prípade blokuje alebo aktivuje funkciu dekodéru. Ak tento vstup bude na úrovni **log0** aj na oboch výstupoch bude **log0**, bez ohľadu na vstup **A**.

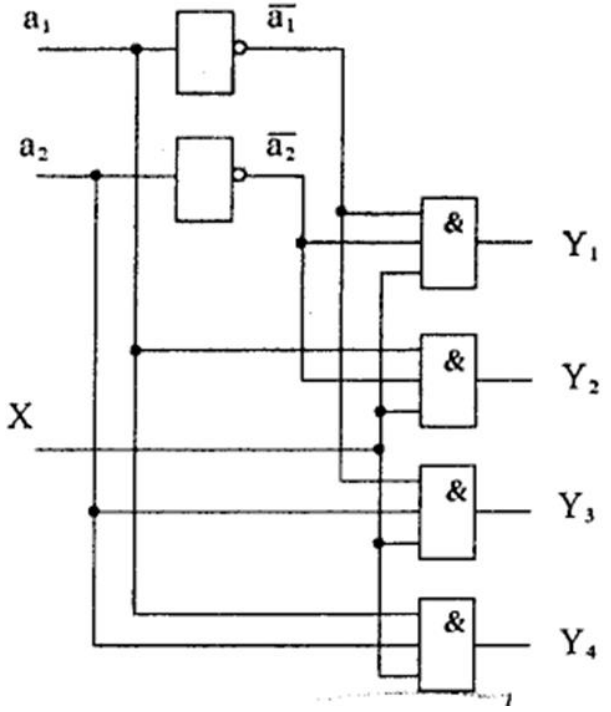
V prípade **D = log1** obvod sa chová ako dekodér. To platí obecne aj pre dekodéry s viac výstupmi a adresovými vstupmi, až na to, že logická úroveň na aktivovanom vstupe nezávisí na úrovni na dátovom vstupe **D**.

Ako demultiplexor môžeme teda použiť dekodér binárneho kódu na kód **1 z n** za predpokladu, že obvod má vstup „uvoľnenia“ na ktorý pripojíme multiplexovaný signál.

Zapojenie dekodéru z prirodzeného dvojkového kódu na kód jedna zo štyroch



B	A	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0



A2	A1	Y1	Y2	Y3	Y4
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

Prevod dekadického čísla do kódu BCD

BCD znamená binárne kódované desiatkové číslo. Pre každú číslicu dekadickej číselnej sústavy pripadajú štyri bity, do ktorých sa zapisuje binárna hodnota danej číslice.

Úloha: *Preveďte dekadické čísla 546 a 930 do BCD kódu*

546D = 0101|0100|0110 BCD

930D = 1001|0011|0000 BCD

Prevod BCD do dekadickej číselnej sústavy

BCD prevádzame do dekadickej číselnej sústavy tak, že všetky štvorice bitov vyjadríme v dekadickej číselnej sústave a tak dostaneme desiatkovú hodnotu.

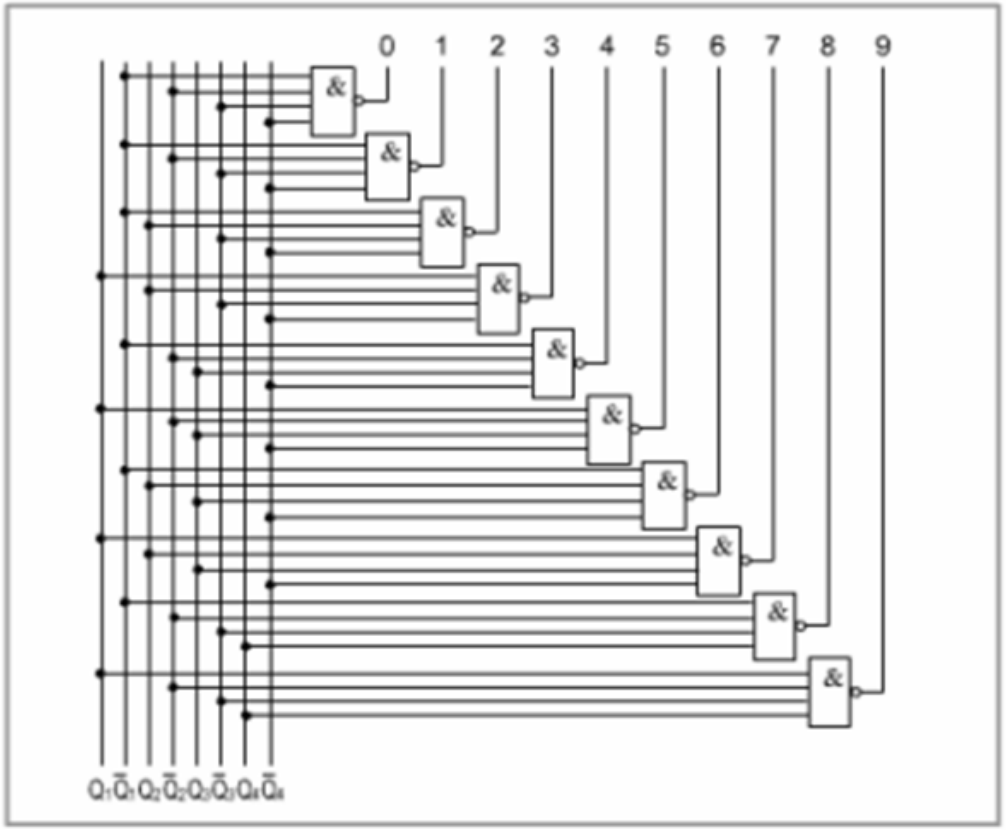
Úloha: *Preveďte dané BCD kódy dekadickej číselnej sústavy.*

0011|1000|0111 - BCD = 387D

0001|0010|0100 - BCD = 124D

Prevodníky kódu

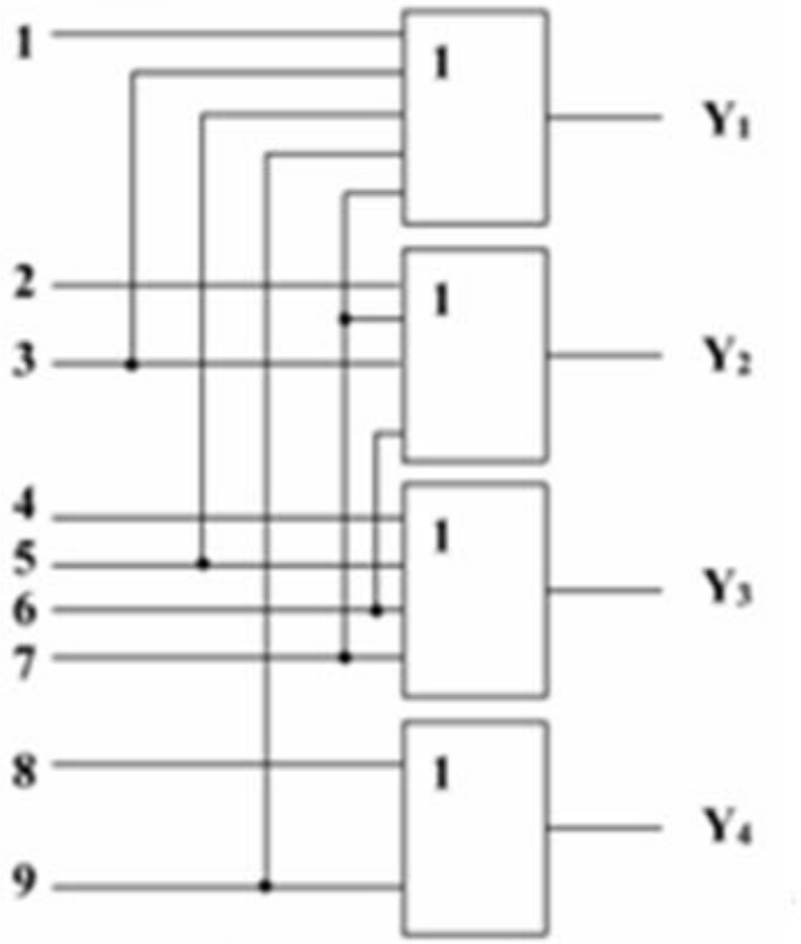
Číslicová technika spracúva informácie v dvojkovej sústave. Človek je ale bežne zvyknutý myslieť v desiatkovej sústave. Je preto občas dôležité použiť logické obvody, ktoré prevedú dvojkový kód na desiatkový.
Na obrázku je príklad zariadenia ktoré prevádza **BCD** kód na desiatkový. Zo vstupných premenných je možné pomocou logických členov **NAND** dekódovať jednotlivé číslice 0 až 9. K ujasneniu funkcie zapojenia si zvolíme číslicu 1. Jej dvojková reprezentácia je 0001.



Dekóder z kódu BCD na 1 z10

Prevod desiatkového kódu na kód BCD

Zapojenie má význam pre zariadenie pre vstup dát napr. numerická klávesnica a pod. Ak sú vstupy otvorené, t.j. na vstupe je log 0, sú na výstupoch $\mathbf{Y_1}$ až $\mathbf{Y_4}$ logické hodnoty 0. Ak stlačíme napr. tlačidlo 7 dostane sa na príslušné vstupy logických členov **OR** hodnota 1 a na výstupnej strane prevodníka sa nastaví nasledujúca informácia:
$\mathbf{Y_1=1, Y_2=1, Y_3=1, Y_4=0=1110=7.}$
Pri tomto zapojení musíme zaistiť, aby sa nedalo stlačiť viacero tlačidiel naraz, môžeme zadať len jednoznačnú informáciu.



Mux_DMux [41]

DEC	HEX	GRAY	BIN	2 KOMPL.	+3	BCD	AIKEN	JOHNSON	7SEGM.	2 z 5	1 z 10
		DCBA	DCBA	BIN + 1	DCBA	DCBA	DCBA	EDCBA	abcdefg	43210	9876543210
0	0	0000	0000	.00000	0011	0000	0000	00000	0000001	00011	0000000001
1	1	0001	0001	11111	0100	0001	0001	00001	1001111	00101	0000000010
2	2	0011	0010	11110	0101	0010	0010	00011	0010010	00110	0000000100
3	3	0010	0011	11101	0110	0011	0011	00111	0000110	01001	0000001000
4	4	0110	0100	11100	0111	0100	0100	01111	1001100	01010	0000010000
5	5	0111	0101	11011	1000	0101	1011	11111	0100100	01100	0000100000
6	6	0101	0110	11010	1001	.0110	1100	11110	0100000	10001	0001000000
7	7	0100	0111	11001	1010	0111	1101	11100	0001111	10010	0010000000
8	8	1100	1000	11000	1011	1000	1110	11000	0000000	10100	0100000000
9	9	1101	1001	10111	1100	1001	1111	10000	0000100	11000	1000000000
10	A	1111	1010	10110					0001000		
11	B	1110	1011	10101					1100000		
12	C	1010	1100	10100					0110001		
13	D	1011	1101	10011					1000010		
14	E	1001	1110	10010					0110000		<="">
15	F	1000	1111	10001					0111000		

Poznámky na záver.

KOMBINAČNÉ LOGICKÉ OBVODY

výstupná veličina je jednoznačne funkciou vstupných veličín

$$y = f(a,b,c\ ..)$$

AND, OR, XOR ...

čiže zmena výstupného signálu môže nastať až po komutácii na vstupe, ale urobí sa to až po určitom čase Δt,

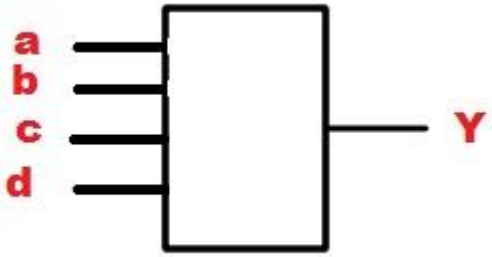
čiže ak komutácia nastane

v čase T - S_0 dostaneme v čase T+Δt, - S_1 v čase T+2Δt

atď.

Je akákoľvek úloha riešiteľná?

Nech je tabuľkou definovaný súbor výstupných hodnôt (náhodne) n - vstupového logického obvodu:



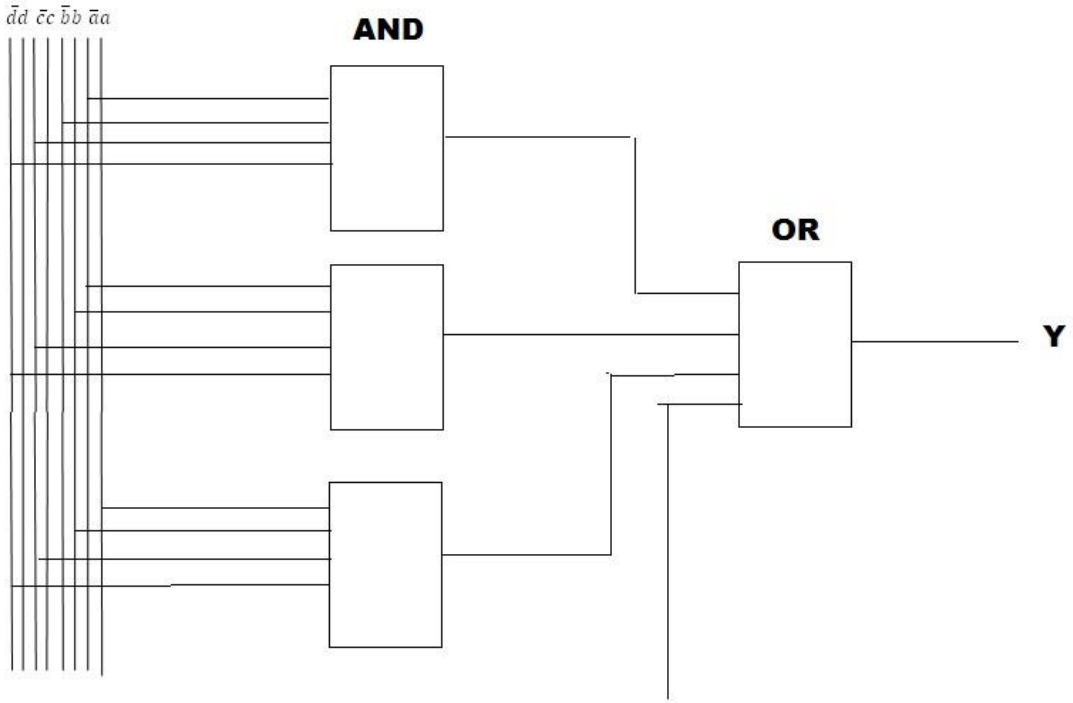
d	c	b	a	Y
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
atď.				**atď.**

Úplná normálna disjunktívna forma = UNDF
tejto funkcie (Y) sa dá napísať v tvare

UNDF(Y) = $\overline{d}\,\overline{c}\,\overline{b}\,\overline{a}$ **+** $\overline{d}\,\overline{c}\,\overline{b}$ **a +** $\overline{d}\,\overline{c}$ **b** $\overline{a}$ **+** $\overline{d}\,\overline{c}$ **b a + ...**

Hľadaná UNDF je súčtom týchto elementárnych súčinových členov v prípade, že Y(a,b,c,d) =1

Konštrukcia hypotetického zariadenia



zbernica

ďalšie členy ...

potreba ďalšej optimalizácie

prílišná optimalizácia = „riziká”