2 DIGITALNA ELEKTRONSKA VEZJA

2.1 ZNAČILNOSTI DIGITALNE TEHNIKE

Digitalni zapis podatka ali informacije je v elektronskih vezjih lahko predstavljen na več načinov (kodov), vendar je vsem skupno, da so napetostni (lahko tudi frekvenčni, fazni, impulzno širinski,..) signali v diskretni obliki. Elektronska vezja najzanesljiveje vzdržujejo dva skrajna nivoja (1,0), zato je zapis signalov podatkov v digitalni obliki najprimernejši za obdelavo. Naše okolje in različne lastnosti ki ga opisujejo, pa so analognega značaja, zato pogosto pretvorimo merjene veličine v digitalno obliko, katera omogoča s pomočjo digitalnih vezij kvalitetnejšo obdelavo, arhiviranje, prenos in druge postopke. Zanimiv primer, kjer se kaže velikanska moč digitalne tehnike je prenos analognih signalov brez vpliva šuma ali motenj, ki se pojavijo na prenosni poti. Poleg tega je pri oslabljenem digitalnem signalu vedno možna rekonstrukcija in ojačitev, kar omogoča prenos na velike razdalje brez napak (npr. internet). Vendar za človeka informacije v tej obliki pogostokrat niso primerne, zato je potreben prikaz v analogni oz. drugi primerni obliki, kar zahteva ponovno pretvorbo. Sodobna digitalna vezja omogočajo izvajanje potrebnih postopkov na različne načine, kateri se s pomočjo sodobne programske opreme, vedno bolj približujejejo človekovi komunikaciji (npr. slika, razpoznava, analiza in sinteza govora,..). Za razumevanje, analizo in sintezo kompleksnih digitalnih vezij je poznavanje temeljnih pravil in komponent digitalne tehnike nujno.

2.2 OSNOVE DIGITALNE TEHNIKE

Za zapis v digitalni obliki so najznačilnejše sledeče lastnosti:

- ✓ Podatki so v diskretni obliki (1-0; nizek nivo-visok nivo,…)
- ✓ Podatki so zapisani v obliki različnih kodov
- ✓ Operacije med podatki se izvajajo po pravilih Boolove algebre
- ✓ Podatki so lahko različne dolžine.
- ✓ Podatki so lahko zapisani v različnem formatu
- ✓ Nad podatki je možno izvajati logične in aritmetične operacije
- ✓ Podatke je možno arhivirati

2.2.1 ŠTEVILSKI SISTEMI

- Desetiški: uteži→…10^2 10^1 10^0,10^{-1} 10^{-2},….nabor znakov→ { *0,1,2,3,4,5,6,7,8,9* }
- Dvojiški: uteži→…2^2 2^1 2^0, 2^{-1} 2^{-2} 2^{-3},….nabor znakov→ { *1,0* }
- Oktalni: uteži→…8^2 8^1 8^0, 8^{-1} 8^{-2} 8^{-3},….nabor znakov→ { *0,1,2,3,4,5,6,7* }
- Šestnajstiški: uteži→…16^2 16^1 16^0,16^{-1} 16^{-2} ….→ { 0,*1,2,3,4,5,6,7,8,9,A,B,C,D,E,F* }

Komplementiranje števil

Glede na to, da v digitalnih vezjih direktnega odštevalnika ne poznamo , je potrebno operacijo odštevanja opraviti s pomočjo seštevalnika. V ta namen uporabljamo postopek za izračunavanje komplementa danega števila. Pravilo je , da mora biti vsota danega števila in njegovega komplementa vedno enaka 0. V bistvu predstavlja komplement danega števila po absolutni vrednosti enako število, vendar z nasprotnim predznakom in ga lahko izračunamo v vsakem sestavu. Vendar pa predstavlja komplementiranje pomembno vlogo le v binarnem sestavu, kjer je izvajanje odštevanja izvedeno v bistvu s pomočjo seštevanja ob predhodnem postopku komplementiranja števila katerega odštevamo. Postopek komplementiranja v dvojiškem sestavu je dvofazen, vendar enostavno izvedljiv. Najprej je potrebno posamezne bite binarnega števila invertirati (enojni komplement), nato pa na najmanj uteženem mestu prišteti še 1 in dobljena vsota predstavlja dvojni komplement danega števila.

Primer:

*Kolikšna je razlika med števili **A=1001010** in **B=1010011**?*
Dvojni komplement števila B je 0101101.

Izračun:

Število A	*1001010*
2" komplement števila B	*+ 0101101*
razlika med števili je:	*1110111*

Pretvarjanje števil iz enega sestava v drugega

Število v drugem sestavu pretvorimo v desetiškega tako, da enostavno seštejemo posamezne produkte števila in pripadajoče »uteži«, kot ponazarjajo sledeči primeri :

$184{,}6_{(10)} = 1\bullet10^2+8\bullet10^1+4\bullet10^0+6\bullet10^{-1}=\underline{184{,}6_{(10)}}$
$1o1{,}1_{(2)} = 1\bullet2^2+0\bullet2^1+1\bullet2^0+1\bullet2^{-1}=\ldots\ldots$
$361{,}4_{(8)} = 3\bullet8^2+6\bullet8^1+1\bullet8^0+4\bullet8^{-1}=\ldots\ldots..$
$B65F{,}A3_{(16)} = 11\bullet16^3+6\bullet16^2+5\bullet16^1+15\bullet16^0+10\bullet16^{-1}+3\bullet16^{-2}=\ldots\ldots$

Pretvorbo desetiškega števila v drugi sestav izvajamo v dveh delih tako, da najprej pretvorimo celoštevilčni del in nato še decimalni del števila in sicer:

- **Celoštevilčni del** pretvorimo tako, da <u>celi del</u> števila in nastale količnike delimo z osnovo željenega sestava tolikokrat, kolikokrat so količniki deljivi. Pripadajoči ostanki pri posameznih delitvah nam v naraščajočem vrstnem redu uteži predstavljajo celoštevilčni del v novem sestavu.
- **Decimalni del** pretvorimo tako, da decimalni del števila (oz. nastale produkte*, ki jih zmanjšamo za celoštevilčni del) množimo z osnovo in pri tem nam celoštevilčni del novega produkta pomeni <u>del števila v novem sestavu</u> na ustreznem utežnem mestu. Produkt nato zmanjšamo za celoštevilčni del (če je!), nato pa množenje ponavljamo tolikokrat, dokler je ostanek (odvisno od natančnosti pretvorbe). Če celoštevilčnega dela ni, pišemo v novem sestavu na tem utežnem mestu »0« in ponovno množimo decimalni del z osnovo. Vrstni red utežnih mest je pri tej pretvorbi ravno nasproten , kot pri pretvorbi celega dela števila.

Primer:
Desetiško število 41, 6875 želimo pretvoriti v dvojiško.

Pretvorba za celoštevilčni del
$41_{(10)} = \ldots_{(2)}$

smer naraščanja uteži ↓

*41:2=20+....→**1***
*20:2 =10+....→**0***
*10:2 =5 +....→**0***
*5:2 =2 +....→**1***
*2:2 =1 +....→**0***
*1:2= 0 +....→**1***

<u>*Rezultat:*</u>
$41_{(10)} = \underline{\mathbf{101001}}_{(2)}$

Pretvorba za decimalni del
$0{,}6875_{(10)} = \ldots_{(2)}$

smer naraščanja uteži ↑

0, 6875• 2=0,3750 +…**1**
0,3750 • 2=0,7500 +…**0**
0, 7500• 2=0,5000 +…**1**
0, 500 • 2=0,0000 +…**1**

<u>*Rezultat:*</u>
$0{,}6875_{(10)} = \underline{\mathbf{0{,}1011}}_{(2)}$

Rezultat je torej: $41_{(10)} = \underline{\mathbf{101001{,}1011}}_{(2)}$

Podobno kot pretvarjamo desetiška števila v dvojiški (binarni) sestav lahko pretvarjamo v osmiški oz. šestnajstiški (heksadecimalni) sestav. Bistvena razlika je v ostanku oz. celoštevilčnem delu, kateri je v teh primerih seveda drugačen in ga sestavljajo znaki iz novega sestava

(**za osmiški sestav:**→ *0,1,…7 oziroma* **šestnjstiški sestav:**→ *0,1,…9,A,…E,F).*

Pravilo za smer naraščanja uteži je enako kot pri pretvorbi v binarno število.

Pogostokrat pa pri digitalnih sistemih in računalnikih poteka pretvorba med števili iz binarnega, heksadecimalnega ali števili oktalnega sestava. V tem primeru je postopek pretvorbe enostavnejši, ker pripadajo vsaki heksadecimalni enoti skupine po štiri oz. oktalni po tri binarne enote.

Primer:
Binarno število $(10110001101011{,}111100000110)_{(2)}$ *želimo pretvoriti 8″ oz. 16″ sestav.*

Za pretvorbo v 8 "sestav, *razdelimo število od decimalne vejice na vsako stran na skupine po tri binarne enote:*

010 110 001 101 011,111 100 000 110
2 6 1 5 3 , 7 4 0 6 ⇒ (26153,7406)(8)

Za pretvorbo v 16 "sestav, *razdelimo število od decimalne vejice na vsako stran na skupine po štiri binarne enote,:*

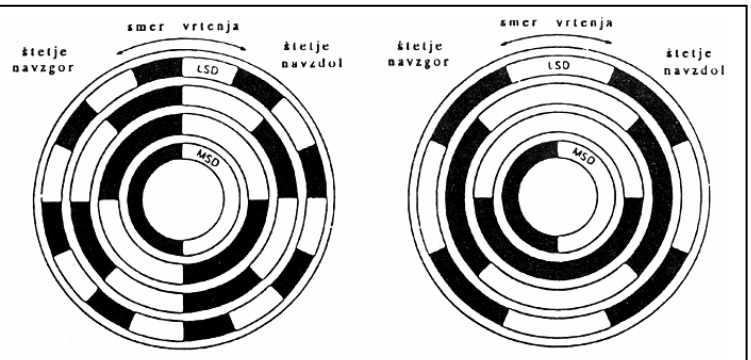
0010 1100 0110 1011,1111 0000 0110
2 C 6 B F 0 6 ⇒ (2C6B,F06)(16)

2.2.2 KODIRANJE, VRSTE IN ZNAČILNOSTI KODOV

Kodiranje je postopek pri katerem s pomočjo črk in nekaterih posebnih znakov po določenem pravilu pretvorimo informacijo (podatek) iz ene oblike v drugo. Cilj tega postopka je lahko različen glede na namen (zajemanje informacije, digitalna obdelava signalov, zaščita podatkov, …), vendar je pri digitalnih vezjih kodiranje predpogoj za katerokoli logično ali aritmetično operacijo. Večinoma je potrebno kodirati decimalna števila, črke, značilnejše in namenske funkcije, ki jih človek enostavno razume, digitalni sistem pa ne. Kodiranje je neke vrste prevajanje na nivo strojnega jezika. Glede na področje uporabe, razlikujemo več vrst kodov, kateri s svojimi različnimi značilnostmi omogočajo dodatne funkcije (npr. enostavnejše procesiranje, enostavno tvorjenje komplementa (Excess-3), preverjanje pravilnosti prenosa,…).

Kodiranje decimalnih števil

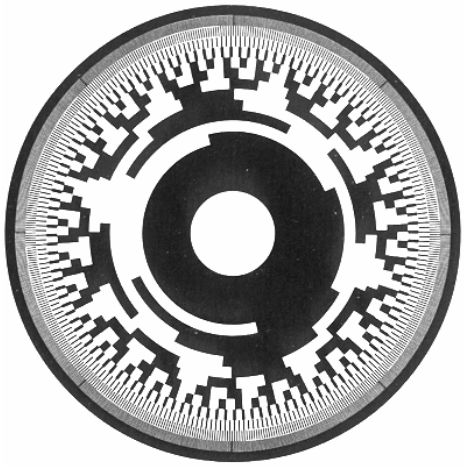
Decimalna števila so lahko kodirana z BCD kodi (*Binary Coded Digit*), ki so lahko utežnostni, neutežnostni (enokoračni, reflektivni) ali pa s kodi, za katere je značilno, da so istočasno enokoračni in reflektivni (npr. Grayev kod). Enokoračno pomeni, da se pri prehodu števila spremeni samo en bit v kombinaciji, reflektivno pa, da se kombinacije (besede), ki so enako oddaljene od horizontalne simetrale razlikujejo vedno le za en bit .



Naravni 8-4-2-1 kod *Gray-ev kod*

Slika 2.1: Kodirni plošči za 4-bitni podatek

Enokoračni kod se uporablja npr. pri digitalnih pomičnih merilih, merilnih letvah za obdelovalne stroje, dajalnikih kotne pozicije-enkoderjih, kar omogoča večjo zanesljivost in enostavno kodiranje-dekodiranje s pomočjo logičnih operacij. Za večjo zanesljivost je potrebno dodati še en bit za kontrolo paritete, s čimer je možno sproti preverjati pravilnost prenosa. Drugačno možnost »varovanja« nudi npr. Hamming-ov kod za katerega je značilno, da morajo vsaki spremembi na vhodu, slediti spremembe stanj najmanj treh od sedmih bitov. Na ta način sta zanesljivo prepoznani dve istočasni morebitni napaki.



Slika 2.2: Praktični primer kodirne plošče za optični dajalnik kotne pozicije (Gray-ev kod)

Primerjava značilnejših kodov

decimalno število	BCD koda				
	8 4 2 1	2 4 2 1	4 2 2 1	8 4-2-1*	Bikvinarna
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0 0 0 1
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 1 1	0 1 0 0 0 1 0
2	0 0 1 0	0 0 1 0	0 0 1 0	0 1 1 0	0 1 0 0 1 0 0
3	0 0 1 1	0 0 1 1	0 0 1 1	0 1 0 1	0 1 0 1 0 0 0
4	0 1 0 0	0 1 0 0	1 0 0 0	0 1 0 0	0 1 1 0 0 0 0
5	0 1 0 1	1 0 1 1	0 1 1 1	1 0 1 1	1 0 0 0 0 0 1
6	0 1 1 0	1 1 0 0	1 1 0 0	1 0 1 0	1 0 0 0 0 1 0
7	0 1 1 1	1 1 0 1	1 1 0 1	1 0 0 1	1 0 0 0 1 0 0
8	1 0 0 0	1 1 1 0	1 1 1 0	1 0 0 0	1 0 0 1 0 0 0
9	1 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 0 1 0 0 0 0

Nekatere vrste različno uteženih BCD kodov
** pomišljaj predstavlja predznak » minus«*

Binarna koda	Gray-eva koda
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

Primerjava Gray-evega koda s čistim binarnim kodom

Kodiranje alfanumeričnih znakov

Za kodiraje alfanumeričnih znakov je v uporabi ASCII (American Standard Code for Information Interchange), katera je 7-bitna in omogoča kodiranje 128 znakov. Ob dodatku še enega bita (8-bitov), omogoča tudi parnost in s tem možnost odkrivanja napak. Kodno tabelo najdemo v elektrotehniškem priročniku, podrobnejši postopki pa že presegajo osnovni uvod v digitalno tehniko.

2.2.3 PRAVILA BOOLOVE ALGEBRE

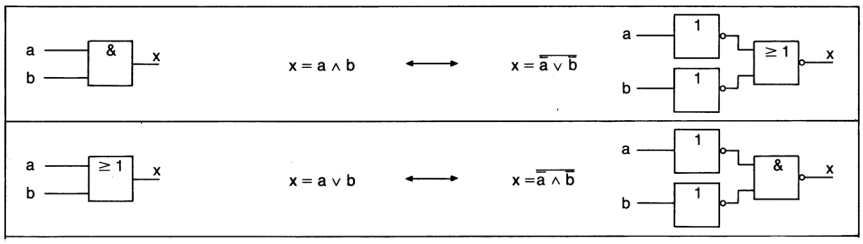
Zanimivo je, da je angleški filozof George Boole, že okoli leta 1850 pisal članke o matematični analizi logike, vendar so šele desetletja pozneje opazili pomembnost njegovega dela. Leta 1937 je namreč C.E. Shannon uporabil Boolove ideje pri poenostavljanju preklopnih vezij v telefoniji. Od takrat se je njegovo delo pod imenom Boolova algebra močno razširilo in pomeni danes osnovo pri konstrukciji in analizi digitalnih sistemov.

Boolova algebra pozna le dve spremenljivki (0,1), nad katerimi lahko ob upoštevanju pravil izvajamo sledeče osnovne operacije:

- Disjunkcija, logična ALI (or, oder)…….+ ; ∨
- Konjunkcija, logični IN (and, und)…• ; **&** ; ∧
- Negacija spremenljivke, funkcije,…….. ⎯ ;

De- Morganov izrek, omogoča pretvarjanje logičnih enačb iz ene oblike v drugo, kar je pri sintezi logičnih vezij večkrat potrebno zaradi poenostavljanja ali preureditve logičnih izrazov. V praksi to pomeni, da je možno isto logično vezje realizirati na več različnih načinov in z minimalnim številom različnih osnovnih gradnikov. Ta vidik je lahko (npr. pri serijski proizvodnji vezij) zaradi zanesljivosti in znižanja stroškov bistvenega pomena (nabava velike količine → nižja cena).

- Dvojna negacija spremenljivke
- Dvojna negacija konjunkcije
- Dvojna negacija disjunkcije



Minterm-m_i je logični produkt vhodnih spremenljivk pri (***i***-ti) kombinaciji:

$$m_i = x_1 \bullet x_2 \bullet \ldots \bullet x_n$$

Maksterm-M_i je logična vsota vhodnih spremenljivk pri (***i***-ti) kombinaciji:

$$M_i = x_1 + x_2 + \ldots + x_n$$

Normalna disjunktivna oblika:

$$f(x_1, x_2, \cdots x_n,) = \sum_{i=1}^{2^n} \alpha_i \bullet m_i = \alpha_1 \bullet m_1 + \alpha_2 \bullet m_2 + \ldots + \alpha_n \bullet m_n$$

Disjunktivna oblika logične funkcije zajema **logično vsoto vseh mintermov** pri katerih je funkcija v stanju »**1**«

Normalna konjunktivna oblika:

$$f(x_1, x_2, \cdots x_n,) = \prod_{i=1}^{2^n} (\alpha_i + M_i) = (\alpha_1 + M_1) \bullet (\alpha_2 + M_2) \bullet \ldots \bullet (\alpha_n + M_n)$$

Konjunktivna oblika logične funkcije predstavlja **logični produkt vseh mintermov** pri katerih je funkcija v stanju »**0**«

V praksi se največkrat poslužujemo disjunktivne oblike logične funkcije, ker jo običajno lažje zapišemo na podlagi predstavitve problema (logična tabela, algoritem,časovni diagram,…).

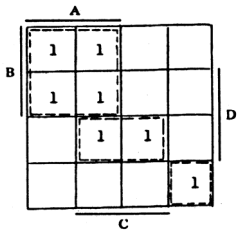
Npr.: Izhodna funkcija bo aktivna pri prvi vhodni kombinaciji ALI pri drugi kombinaciji ALI … (vse ostale kombinacije, pri katerih mora biti funkcija aktivna)

Seveda v funkciji odpadejo vsi členi pri katerih funkcija ni aktivna (α_i=0), kar zapis običajno močno skrajša.

Poenostavljanje logičnih enačb

Na podlagi pravilnostne tabele, lahko zapišemo posamezne kombinacije –minterme, ki jih povežemo z ali funkcijo (disjunktivna oblika). Kljub temu, da v enačbi ostanejo samo mintermi pri katerih zavzame funkcija stanje »1«, je logični izraz največkrat kompliciran in bi zahteval veliko logičnih komponent. Zato je potreben še postopek poenostavitve-minimizacije logičnih izrazov, s katerim dosežemo željeno funkcijo z minimalnim številom komponent. Za funkcije s do pet vhodnimi spremenljivkami lahko uporabimo grafične metode poenostavljanja kot sta npr. Veitchev ali Karnoughov diagram. Pri teh metodah povezujemo 2,4,8,16 ali 32 minteremov v skupine, ki imajo eno ali več spremenljivk v komplementarni obliki.

$f = AB\overline{CD} + ABCD + A\overline{B}CD + \overline{A}B\overline{CD} + AB\overline{C}D + ABC\overline{D} + \overline{A}\overline{B}CD$



Poenostavljena funkcija je:

$$f = AB + \overline{B}CD + \overline{A}\overline{B}\overline{CD}$$

Slika 2.3: Primer poenostavitve logične funkcije z Veitchevim diagramom

Za funkcije z več spremenljivkami pa uporabljamo tabelarične metode.Taka je Queen-Mc Clusky-jeva metoda povezovanja oz. izločevanja členov, ki se podvajajo, ne nastopajo ali se »pokrivajo«. Uporabljajo jo nekateri računalniški programi (npr.EWB).

Poenostavljene logične izraze po potrebi s pomočjo De-Morganovega izreka in drugih pravil Boolove algebre še primerno preoblikujemo, da je vezje mogoče realizirati z izbranimi digitalnimi vezji, ki so na razpolago (npr. NAND, NOR,…). Potrebno logično funkcijo je namreč mogoče realizirati na več načinov z različnimi vrstami in tipi digitalnih komponent.

Seveda lahko iste metode uporabimo pri programiranju prosto programirljivih krmilnikov, kjer na ta način prihranimo pomnilniški prostor, poenostavimo programiranje in povečamo preglednost programa.

Postopke minimizacije in preoblikovanja logičnih funkcij si lahko interaktivno pojasnite tudi na sledeči spletni strani: http://tech-www.informatik.uni-hamburg.de/applets/kvd/tutorial-fnh/index.html

2.2.4 SIMBOLI IN OZNAČBE LOGIČNIH OPERATORJEV

Za grafično podajanje značilnejših digitalnih funkcij se uporabljajo različni simboli, katerih oblika je odvisna od področja uporabe in kar povzroča težave pri analizi oz. sintezi vezij. V tej smeri seveda poteka poenotenje, res zelo počasi, s sprejetjem standardov IEC 617-12 in 113-7 (DIN 40900 T12) jih uporabljamo tudi pri nas. Ameriške in ostale simbole (DIN 40700) je seveda potrebno poznati zaradi razumevanja, vendar jih ne smemo uporabljati pri izdelavi novih dokumentacij oz. načrtov.

Za kompleksnejše logične funkcije predpisuje standard osnovno obliko in pravila za označevanje simbolov, pri čemer so medsebojne povezave med vhodi, izhodi ali vhodi in izhodi podane z odvisnostno označbo. Podrobneje je ta standardizacija opisana v Priročniku za elektrotehniko in elektroniko-Friedrich v poglavju o grafičnih simbolih za binarne elemente.

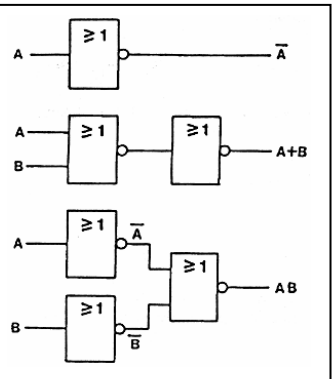
SIMBOL	OZNAKA	LOG. FUNKCIJA	
a, b, &	IN	$x = a \wedge b$	a b x 0 0 0 0 1 0 1 0 0 1 1 1
a, b, ≥1	ALI	$x = a \vee b$	a b x 0 0 0 0 1 1 1 0 1 1 1 1
a, 1	NEGACIJA	$x = \bar{a}$	a x 0 1 1 0
a, b, &	NAND	$x = \overline{a \wedge b}$	a b x 0 0 1 0 1 1 1 0 1 1 1 0
a, b, ≥1	NOR	$x = \overline{a \vee b}$	a b x 0 0 1 0 1 0 1 0 0 1 1 0
a, b, =	ekvivalenca negirana antivalenca	$x = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$	a b x 0 0 1 0 1 0 1 0 0 1 1 1
a, b, =1	antivalenca XOR	$x = (\bar{a} \wedge b) \vee (a \wedge \bar{b})$	a b x 0 0 0 0 1 1 1 0 1 1 1 0

Slika 2.4: Predpisani simboli za logične operatorje

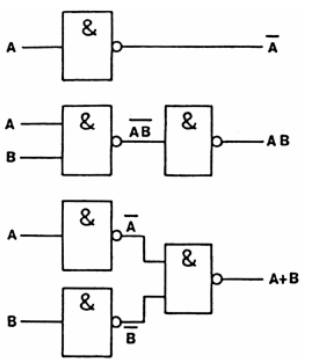
Osnovni operatorji logičnih vezij

Osnovni operatorji, pripadajoči simboli, logične enačbe in pravilnostne tabele za logična-kombinacijska vezja so prikazani na levi sliki.
Najbolj uporabljivi so operatorji NAND (*Sheffer*), NOR (*Pierce*), XOR, kajti z njimi je možno realizirati vse ostale operatorje vključno z negacijo.

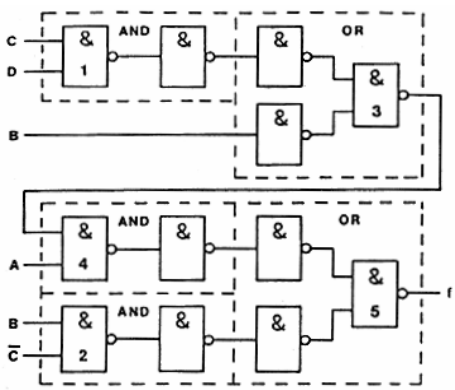
Značilnejši primeri in izvedbe kombinacijskih funkcij z NAND/NOR gradniki so na slikah spodaj.



Slika 2.5: Izvedba osnovnih funkcij z NOR operatorji



Slika 2.6: Izvedba osnovnih funkcij z NAND operatorji



Slika 2.7: Izvedba kombinacijskega vezja z NAND operatorji

Z različnimi medsebojnimi povezavami osnovnih operatorjev so realizirana tudi standardna kombinacijska vezja (kodirnik, dekodirnik, seštevalnik, primerjalnik, multipleksor,..), ki imajo prav tako standardizirane specifične simbole.

Ponovite in utrdite !

Analiza notranjih povezav pri standardnih kombinacijskih vezjih (kodirnik, dekodirnik, prekodirnik, multiplekser, demultiplekser, binarni primerjalnik, polovični in popolni seštevalnik,...