Marijo Nižetić

Laboratorijske vježbe
Digitalni sklopovi
simulacijskim alatom LogiSim 2.7.1.

prema knjigama:

1. Charles W. Kann III, *Digital Circuit Projects*
2. M. Morris Mano, Michael D. Ciletti, *Digital Design With an Introduction to the Verilog HDL*
3. Stanko Paunović: *Digitalna elektronika 1*
4. George Self, *Lab Manual For Exploring Digital Logic with Logisim*
5. George Self, *Exploring Digital Logic With Logisim*
6. Aleksandar Szabo, *Impulsna i digitalna elektronika I i II*, Školski centar "Ruđer Bošković", Zagreb, Getaldićeva bb, 1973.
7. Daniel J. Tylaysky, *Digital Design for the Laboratory*
8. www.learnabout-electronics.org, *Digital-Electronics-Module-02.pdf*

Primjeri

Sadržaj

0. 0. PREDGOVOR

Ova skripta temelji se na knjigama: Aleksandar Szabo, *Impulsna i digitalna elektronika* I i II, Školski centar "Ruđer Bošković", Zagreb, Getaldićeva bb, 1973. i Stanko Paunović: *Digitalna elektronika 1*, Zagreb, 1995. Materija je organizirana prema nazivima poglavlja iz navedenih knjiga, ali su svi crteži nastali kao rezultat simulacijskih primjera programskim alatom LogiSim 2.7.1. Svaka pa i najmanja sličica (ali ne u ovome *.pdf dokumentu) predstavlja poveznicu (*hyper link*) na simulacijski primjer koji se pokrene klikom miša i nastavlja raditi samostalno omogućujući razne simulacijske inačice istoga sklopa. One se mogu dodatno opremiti sklopovima spomenutoga simulacijskoga alata s ciljem automatizacije rada osnovnoga sklopa (automatsko okidanje raznim frekvencijama takta, automatsko preklapanje izlaza ili ulaza, automatsko zaustavljanje rada simulacijskoga sklopa nakon postignutih, unaprijed postavljenih uvjeta zaustavljanja i sl.).

Osim naziva poglavlja, privremeno sam zadržao slike tranzistorskih i diodnih sklopova (zbog žurbe oko obuke studenata i do izrade vlastitih). Također sam "prisvojio" nazive i sadržaje poglavlja: "Pregled ključnih pojmova" te "Pitanja i zadaci za ponavljanje". Zadržao sam redoslijed ali sam ih sadržajno prilagodio. Dio primjera preuzeo sam iz knjige "Aleksandar Szabo, *Impulsna i digitalna elektronika I i II, Centar odgoja i usmjerenog obrazovanja za elektroniku, preciznu mehaniku i optiku "Ruđer Bošković", Zagreb, Getaldićeva bb, 1973.*" i obradio alatom "SimuLink" na isti način.

Ovoj skripti pridruženi su dodatni sklopovi, jer posjeduju opća svojstva pa se mogu koristiti i u nekim drugim primjenama i primjerima simulacija navedenoga simulacijskoga alata. Ovakvom organizacijom umanjuje se zbunjivanje čitatelja koji nisu svladali osnove kombinacijskih i serijskih logika, a služe upravo za samostalnu nadgradnju znanja čitatelja (studenta).

Logički sklopovi za digitalne sustave mogu biti kombinacijski (*combinational*) ili serijski (*sequential*).

Kombinacijski sklop sastoji od logičkih vrata čiji se izlazi u bilo koje vrijeme određuju iz samo ove kombinacije ulaza. Kombinacijski krug obavlja operaciju koju opisuje skup logičkih Booleovih funkcija. Nasuprot tome, serijski sklopovi, osim logičkih vrata koriste elemente za skladištenje. Njihovi izlazi su funkcija ulaza i stanja elemenata za pohranu.

Kako je stanje elemenata za pohranu funkcija prethodnih ulaza, izlazi iz serijskoga sklopa ne ovise samo o trenutnoj vrijednosti ulaza, nego i o prošlim ulazima, a ponašanje kruga mora se odrediti u vremenskim nizovima ulaza i unutarnjih stanja. Serijski krugovi izgrađeni su od digitalnih sustava.

Dakle, kombinacijski logički sklopovi su logički sklopovi čije izlazno stanje ovisi o trenutnome stanju ulaza, a serijski (*slijedni*) (logički) sklopovi su sklopovi čije izlazno stanje ovisi o trenutnome ulaznome stanju i o prethodnim stanjima. I jedna i druga vrsta sklopova mijenjaju svoja stanja u diskretnim vremenskim trenutcima što ih određuje upravljački takt-impuls. Detaljne opise potražite u mojim materijalima: "Zaštitno kodiranje signala-skripta" i "Laboratorijske vježbe iz kolegija 'Zaštitno kodiranje signala'".

Marijo Nižetić

p.s. Za besplatne simulacijske primjere, obratite mi se na marijo.nizetic@st.t-com.hr

In the case of your interest to obtain simulations present in this material and made by Logisim 2.7.1 tool, don't hesitate contact me on above e-mail.

1. 1. BROJEVNI SUSTAVI I KÔDOVI

1.0. Kreiranje brojevnoga sustava

1.0.1. PORIJEKLO NAZIVA NEKIH BROJEVNIH SUSTAVA:

broj	grčki naziv
0	
1	mona (hena)
2	**binar (di)**
3	terna (tri)
4	tetra (quadri)
5	penta
6	hexa
7	hepta
8	**octa**
9	nona (ennea)

broj	grčki naziv
10	**deca**
11	undeca (hendeca)
12	dodeca
13	trideca
14	tetradeca
15	pentadeca
16	**hexadeca**
17	heptadeca
18	octadeca
19	enneadeca

broj	grčki naziv
20	icosa
30	triaconta
40	tetraconta
50	pentaconta
60	**hexaconta**
70	heptaconta
80	octaconta
90	enneaconta
100	hecto
10000	myria

U SVAKOME brojevnome sustavu, kada se iscrpe osnovne znamenke brojevnoga sustava, sljedeća znamenka je 10 (čita se: "jedan"-"nula").

SUSTAV	BAZA	MOGUĆE ZNAMENKE	OZNAKA ZA DEKADSKI 10_{10}	
Binarni	2	0, 1	1010_2	1010
Oktalni	8	0, 1, 2, 3, 4, 5, 6, 7	12_8	013
Dekadski	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	10_{10}	10
Heksadekadski	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	A_{16}	0xA
Neki novi (trinarni)	**3**	**0, 1, 2**	**21_3**	**21**
Sexagesimal[1]	60	1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60		

1.0.2. PRIMJERI BROJEVNIH SUSTAVA:

Moguće je jednostavno napraviti svoj brojevni sustav pa sljedeća tablica prikazuje tri takva, a zovu se npr. *pentarni* (5 osnovnih znamenaka), *nonarni* (9 osnovnih znamenaka) i *tridekadski* (13 osnovnih znamenka), ali oni imaju samo teorijsko značenje, jer nemaju svoj preslik ni primjenu u stvarnome svijetu.

Naziv sustava	Brojevi rastućim redoslijedom u bilo kojemu brojevnome sustavu																
binarni	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
neki novi baza = 5 *pentarni*	0	1	2	3	4	10	11	12	13	14	20	21	22	23	24	30	31
oktalni	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20
neki novi baza = 9 *nonarni*	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17
dekadski	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
neki novi baza = 13 *tridekadski*	0	1	2	3	4	5	6	7	8	9	A	B	C	10	11	12	13
heksa-dekadski	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Ovaj C++ program: (Sljedeća znamenka je 10), pokazuje da u bilo kojemu brojevnome sustavu - stvarnome ili izmišljenome, kada se pobroje sve osnovne znamenke toga sustava, sljedeće znamenka je 10 (čita se: "jedan"-"nula"!) (vidi izvorni C++ kôd).

1.0.3. PORIJEKLO NAZIVA "*HEXDECIMAL SYSTEM" U ENGLESKOME JEZIKU*

Izraz "*Hexadecimal*" osmislili su 1963. godine u IBM. On je izveden iz grčke riječi *hexi* (šest) i latinske riječi *decem* (deset). Prikladniji izraz koristio bi latinsku riječ *sexa* (six), ali izraz *sexidecimal* ne zvuči previše pristojno.

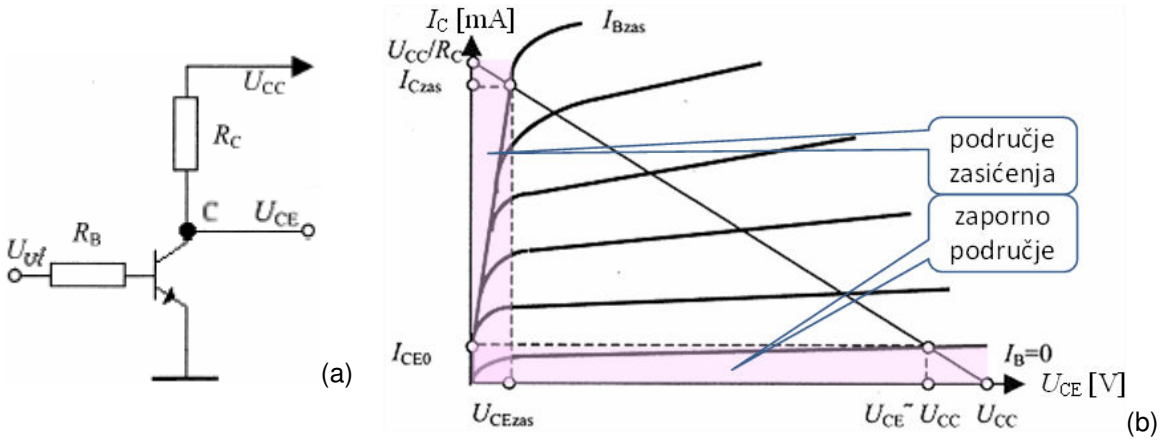
---

[1] Prije 4000 godina Babilonci su razvili *sexagesimal* (latinski "šezdeset") brojevni sustav (baza 60). Svaka znamenka prenosi $\log_2 60 = 5.91$ informacijskih bitova. Na primjer, broj $4000_{10} = 1\ 6\ 40_{60}$ (1 u 3600. stupcu, 6 u 60. stupcu i 40 u 1. stupcu).

1.0.4. TEMELJ DIGITALNE TEHNIKE

Današnja moderna komunikacijska mreža i pridruženi joj sklopovi, u svojim temeljima sadrže kombinacijske i serijske logičke sklopove. Iako teorijske osnove logičkih sklopova sežu sve do kraja pretprošloga stoljeća, svoj pun učinak iskazuju danas u digitaliziranoj integriranoj komunikacijskoj mreži. Približavanjem klasičnih telekomunikacijskih mreža (razvijanih sve do kraja prošloga stoljeća) i računalnih mreža (što su svoj pun zamah razvoja doživjele u suton klasičnih telekomunikacijskih sustava), došlo se do njihove potpune integracije pa danas klasična telekomunikacijska govorna usluga predstavlja samo jednu od usluga u združenoj digitaliziranoj mreži.

Rad digitalnih sklopova zasniva se na radu tranzistora u načinu rada sklopke (*zaporno* područje i područje *zasićenja* u polju izlaznih karakteristika). Tranzistor u ovome načinu rada, ima svoju matematičku podlogu u linearnoj algebri i binarnoj logici, a cjelovita Booleova algebra preslikala se iz područja teorije u područje prakse na učinkovit način. Današnja tehnologija omogućila je široku primjenu logičkih sklopova uz zanemarivu potrošnju energije, malen volumen, visok stupanj integracije, veliku brzinu tada i nisku cijenu. Ovo su zapravo, pravi temelji svih dijelova komunikacijskih mreža i uređaja. Slika prikazuje bipolaran PNP tranzistor u spoju zajedničkoga emitera i pripadne izlazne karakteristike.



Osjenčano područje u polju izlaznih karakteristika uz *y*-os naziva se *područje zasićenja*, jer u tim radnim uvjetima tranzistorom teče maksimalna *struja*. Preslikano u područje digitalne tehnike ovo bi moglo predstavljati logičku *jedinicu* (zatvoren strujni krug kroz tranzistor i struja I_C je maksimalna). Međutim, promatrajući *naponsku razinu* u točki kolektora C ako tranzistor vodi, vidi se da je napon U_{CE} u tome trenutku minimalan pa bi i to mogao biti i jest kriterij za logičku razinu, ali sada *nulu*.

Osjenčano područje u polju istih izlaznih karakteristika ali uz *x*-os, naziva se *zaporno područje*, jer tranzistorom praktički ne teče struja pa ovakav režim rada u digitalnoj tehnici, predstavlja logičku nulu. Ako opet promotrimo što je s *naponom* u tome trenutku u točki kolektora C, on ima maksimalan iznos U_{CC}. Odlučimo li se za takvo promatranje (a to jest slučaj), u točki C (na izlazu) imamo logičku *jedinicu*.

U svakome slučaju, tranzistor se ponaša kao sklopka, ili vodi ili ne vodi struju, odnosno posljedično promatrana izlazna točka C je na naponskoj razini ili nula ili jedan. Kriterij izbora logičke jedinice i logičke nule mogla bi biti minimalna potrošnja struje integriranih sklopova (sklopki) u dinamičkome radu pa to može biti ozbiljan argument pri izboru jednoga ili drugoga kriterija za logičke veličine. U teorijskome smislu točka motrišta je nevažna, ali s praktične točke motrišta, jednostavnije i praktičnije je mjerenje *naponskih razina* od mjerenja *protoka struje* kroz bilo koji sklop. Ova činjenica sugerira nam izbor.

Sklop mora baratati strogo odijeljenim vrijednostima napona između niske i visoke razine, jer bi vrijednosti napona između ovih dviju ekstremnih vrijednosti mogle onesposobiti logički rad sklopa zbog neizvjesnosti pridruživanja napona niske razine nuli ili napona visoke razine jedinici. Metodom dedukcije dolazi se do spoznaje da se u digitalnim sklopovima i uređajima, električne veličine mogu preslikati u znamenke binarnoga brojevnoga sustava. Otuda proizlazi naziv za digitalnu elektroniku, jer riječ *digit* znači prst (znamenka). Zbog toga je za razumijevanje rada digitalnih sklopova, potrebno poznavati *brojevne sustave* i *kôdove* kao temelje baratanja digitalnim podacima.

Naravno, postoje i sklopovi u digitalnoj tehnici koji barataju vrijednostima napona između ovih dvaju ekstrema (0 i 1). Pri tomu se koristi *neizrazita logika* za odluku kojoj ekstremnoj vrijednosti pripada promatrana naponska razina. Ovakve metode pripadaju skupu metoda dekodiranja informacija i donošenju tzv. mekih odluka pripada li primljena očitana razina napona logičkoj nuli ili logičkoj jedinci. Meke odluke dekodiranja dodatno koriste postulate teorije vjerojatnosti i matematičke statistike. Pristup *neizrazitom logikom* obrađuje se u drugome dijelu ovih skripti.

1.1. 1.1. BROJEVNI SUSTAVI

U svakodnevnoj uporabi najviše se koristi *dekadski* brojevni sustav. Razmatranjem najprije, dekadskoga brojevnoga sustava olakšava razumijevanje ostalih brojevnih sustavima koji se koriste u digitalnoj elektronici. Osim binarnoga, to su oktalni i heksadekadski brojevni sustav.

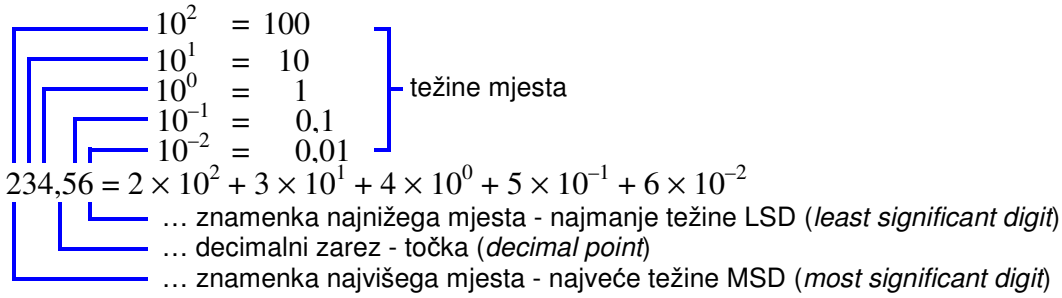
1.1.1. 1.1.1. DEKADSKI BROJEVNI SUSTAV

U bilo kojemu (svakomu) brojevnome sustavu, prva znamenka je 0. Dekadski brojevni sustav ima deset znamenaka: 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9, što znači da se svi brojevi od nula do devet mogu prikazati jednim simbolom (znamenkom). To su jednoznamenkasti brojevi. Najveći broj koji se može napisati jednom znamenkom je devet. Broj deset nema posebnu oznaku pa se piše kombinacijom dviju znamenaka 1 i 0 (ili skraćeno 10 - čita se *deset* za razliku od ostalih brojevnih sustava gdje se ista kombinacija znamenaka čita "*jedan*"-"*nula*"). Dvjema znamenkama mogu se napisati svi brojevi od deset do devedeset devet. To su dvoznamenkasti brojevi.

Za veće brojeve, potrebne su tri ili više znamenaka. Najveći broj u dekadskome sustavu što ga se može napisati s n znamenaka iznosi $10^n–1$. Položaj znamenke u bilo kojemu broju naziva se *brojevno mjesto*. Svako brojevno mjesto ima svoju *vrijednost*, odnosno *težinu*. Težine brojevnih mjesta u dekadskome brojevnome sustavu mogu se prikazati kao *potencije broja deset* (ukupan broj osnovnih znamenaka u sustavu). Zato se kaže da je deset *osnovica* ili *baza* dekadskoga brojevnoga sustava. Najniže cjelobrojno mjesto ima težinu $10^0 = 1$.

Težine viših brojevnih mjesta iznose npr.: $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$, itd. Težine mjesta desno od dekadskoga zareza pišu se redom 10^{-1}, 10^{-2}, 10^{-3}, itd.

0. Primjer: Težine mjesta desno od decimalnoga zareza



U dekadskome brojevnome sustavu brojevi se prikazuju nizom znamenaka koje naznačuju koeficijente kojima se množi baza sustava potencirana pripadnim težinskim mjestom. Pri pisanju brojeva pišu se samo koeficijenti (ne i potencirana baza), a težine mjesta određuju se prema položaju koeficijenata lijevo i/ili desno od decimalnoga zareza. Opći prikaz broja u dekadskome brojevnome sustavu izgleda:

$$X = d_n\times10^n + d_{n-1}\times10^{n-1} + \ldots + d_2x\ 10^2 + d_1\times10^1 + d_0\times10^0 + d_{-1}\times10^{-1} + d_{-2}\times10^{-2} + \ldots + d_{-(m-1)}\times10^{-(m-1)}\times d^{-m}\times10^{-m}$$

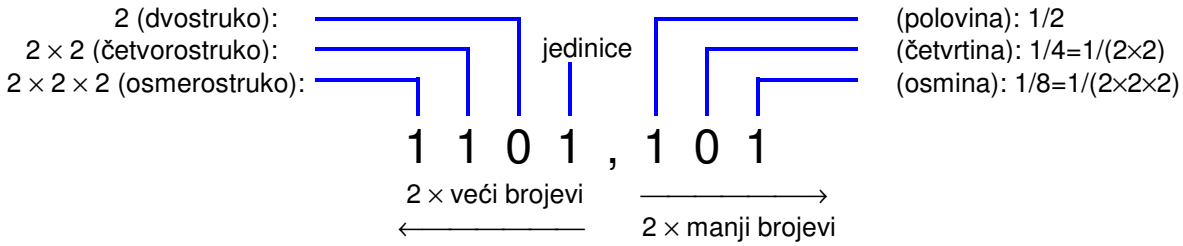
Koeficijenti d, predstavljaju znamenke dekadskoga brojevnoga sustava.

1.1.2. 1.1.2. BINARAN BROJEVNI SUSTAV

Binarna matematika posebna je grana matematike, a bavi se brojevnim sustavom koji sadrži samo dvije znamenke: 0 i 1. Čini se vrlo sputavajuće koristiti samo dvije osnovne znamenke. Međutim, lako je stvoriti elektroničke komponente koje mogu razlikovati dvije naponske razine (a ne deset koliko bi trebalo npr. za dekadski brojevni sustav). Komponente se jeftine, malih su dimenzija i male potrošnje snage.

U dekadskome brojevnome sustavu postoje: jedinice, desetice, stotice, ….

U binarnome sustavu postoje jedinice, dvojke, četvrtice itd. poput sljedećega prikaza:



Ovo znači:

$$1\times8 + 1\times4 + 0\times2 + 1\times1 + 1\times(1/2) + 0\times(1/4) + 1\times(1/8) = 13{,}625 \text{ u dekadskome.}$$

Brojevi se smještaju lijevo ili desno od decimalnoga zareza za naznačiti vrijednosti veće odnosno manje od 1.

Kao što se prije navelo, binaran brojevni sustav ima samo dvije znamenke: 0 i 1. Zbog toga je za pisanje npr. broja 2, potrebno koristiti kombinaciju dviju binarnih znamenaka pa se u binarnome sustavu dekadski ekvivalent broju 2 piše kao 10. Najveći dvoznamenkasti broj u binarnome brojevnome sustavu je 11, što odgovara dekadskome broju 3, a najveći broj što ga se uopće može napisati s n znamenaka iznosi $2^n–1$.

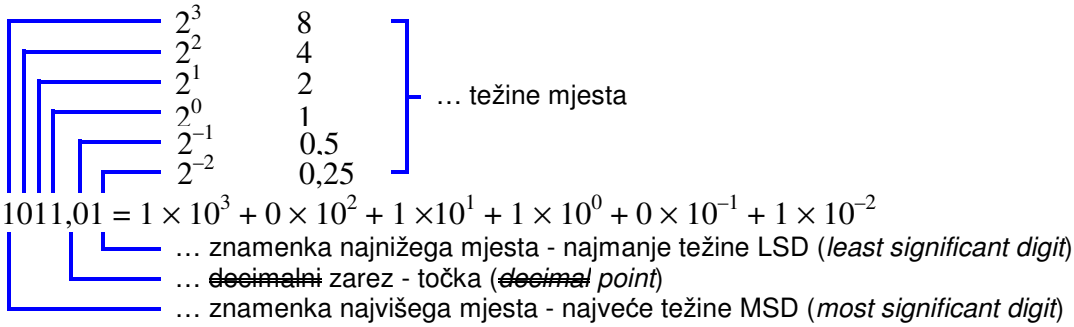
Baza binarnoga brojevnoga sustava je 2. Težine cjelobrojnih mjesta u binarnome brojevnome sustavu su: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, …. Brojevna mjesta desno od binarnoga zareza imaju težine 2^{-1}, 2^{-2}, 2^{-3}, …. Za znamenke binarnoga brojevnoga sustava (binarne znamenke) obično se koristi naziv bît (*binary digit*). Opći prikaz broja u binarnome brojevnome sustavu izgleda:

$$B = b_m\times2^m + b_{m-1}\times2^{m-1} + \ldots + b_1\times2^1 + b_0\times2^0 + b^{-1}\times2^{-1} + \ldots + b_{-(n-1)}\times2^{-(n-1)} + b_{-n}\times2^{-n}$$

Koeficijenti b polinoma B, pripadaju znamenkama binarnoga brojevnoga sustava.

Dakle i u binarnome brojevnome sustavu, brojevi se prikazuju nizom znamenaka koje označavaju koeficijente kojima se množi potencirana baza toga brojevnoga sustava potencijom, a koja odgovara brojevnome mjestu. Isto kao i u dekadskome brojevnome sustavu, pišu se samo koeficijenti polinoma, a težine mjesta određuju se prema mjestu tih koeficijenata.

1. *Primjer: Određivanje težina mjesta prema položaju koeficijenata*

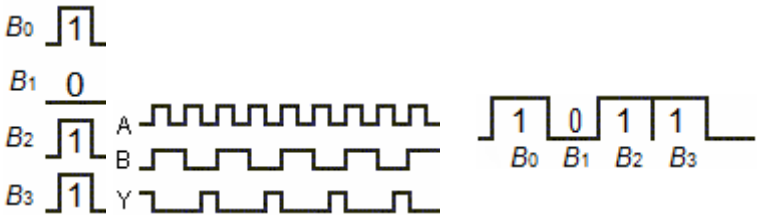


Tablica 1.1. Prikaz prvih 16 brojeva u binarnome i dekadskome brojevnom sustavu

Binaran broj	Dekadski broj	Binaran broj	Dekadski broj
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

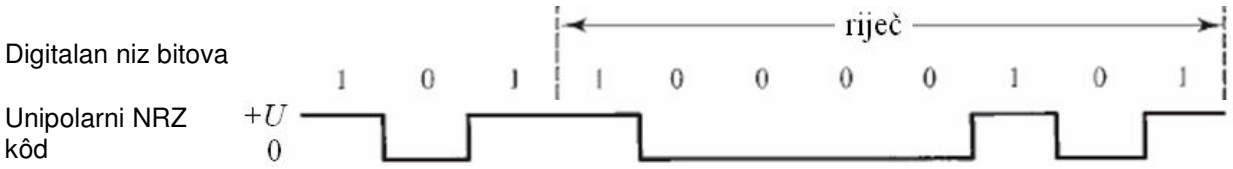
1.1.3. 1.1.3. BINARNI SIGNALI

Signalima u digitalnoj elektronici pridružuju se kombinacije binarnih znamenaka pa se nazivaju *binarni signali*, a mogu biti serijski ili paralelni.

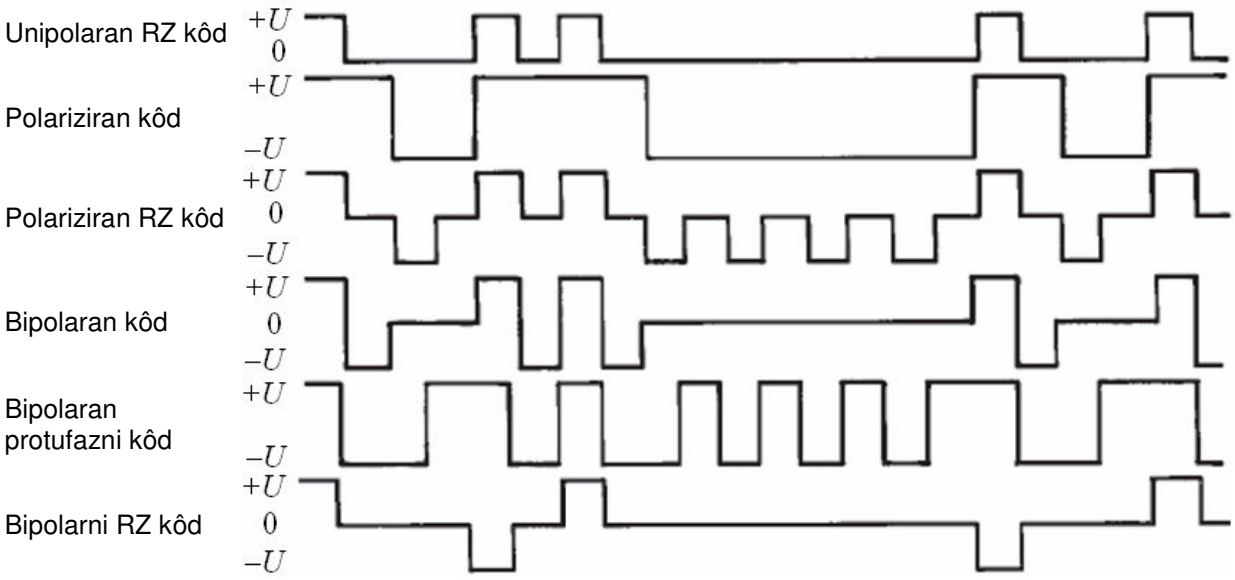


(a) *Paralelni binarni signali* (b) *Serijski binarni signali*

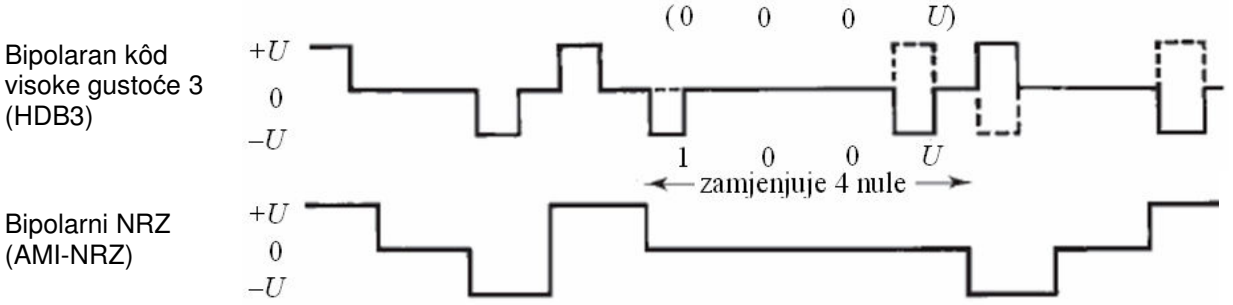
U praksi se koriste različiti valni oblici za predstaviti binarne simbole "0" i "1". Ako se promatra cjelovitost prijenosa digitalnih informacija, ovo postiže pravi učinak. U prisutnosti šuma u kanalu, primjena prikladni(ji)h digitalnih valnih oblika, osigurava se maksimalna razlučivost nula od jedinica. Formati signala (kodova) što se koriste za tu namjenu su: unipolarni kôd bez povratka na nulu, diferencijski unipolarni kôd bez povratka na nulu, bipolarni kôd bez povratka na nulu, bipolarni kôd s povratkom na nulu, Manchester kôd i diferencijski RZ kôd. Sljedeći niz slika prikazuje najpoznatije valne oblike za tu namjenu uz djelomične i kratke opise.



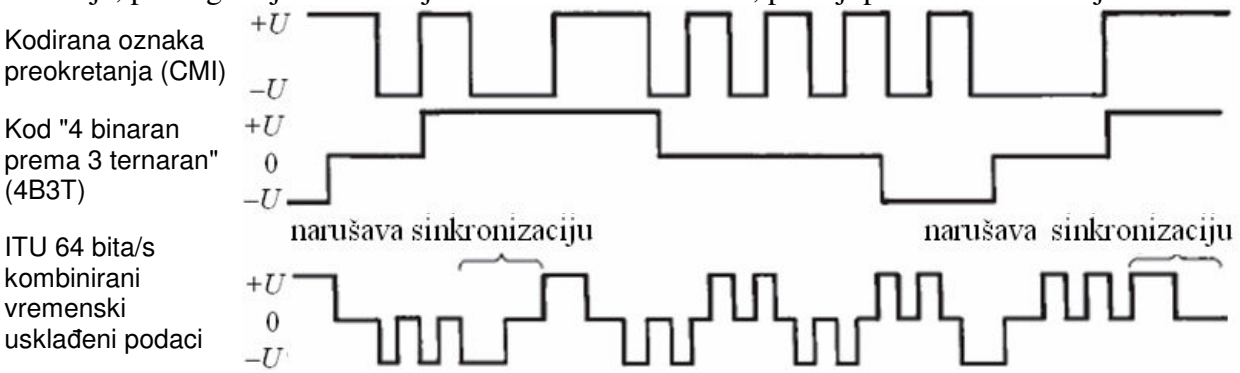
Binarna 1 kodira se impulsom pozitivne razine, a binarna 0 izostankom impulsa. Za dugotrajan niz "1" ili "0", nema prijelaza u digitalnome signalu, pa se gubi informacija o taktu. Zato se ovaj format ne može koristiti za sinkronizaciju između predajnika i prijemnika.



Binarna 1 predstavlja se pozitivnim, a binarna 0 negativnim impulsom. Na sredini svakoga intervala, signal pada na nultu razinu. Kako na sredini svakoga interval postoji prijelaz, sinkronizacija između predajnika i prijemnika lako se održava. Nedostatak su dvostruko brže promjene signala što zahtijeva dvostruko veću propusnost.



Za vrijeme prijenosa jednoga bita razina signala je stalna. Koriste se dvije naponske razine, ali za razliku unipolarnoga NRZ kôda, pozitivan impuls kodira binarnu 1, a negativan impuls kodira binarnu 0. Ovdje, pri dugotrajnome nizu jedinica 1 odnosno nula 0, postoji problem održavanja sinkronizacije.



1.1.4. 1.1.4. PRETVORBA BROJEVA IZMEĐU BINARNOGA I DEKADSKOGA SUSTAVA

Broj iz binarnoga brojevnoga sustava pretvara se u odgovarajući broj dekadskoga sustava tako da se svaka znamenka binarnoga broja pomnoži svojom težinom mjesta, a tako dobiveni iznosi se zbroje.

2. Primjer: Pretvorba binarnog broja 1011001 u dekadski.

$$D = \overset{6\,5\,4\,3\,2\,1\,0}{1011001_2} = 1\times2^6 + 0\times25 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 1\times64 + 1\times16 + 1\times8 + 1\times1 = 89_{10}$$

Indeksi uz brojeve pokazuju u kojemu brojevnome sustavu je broj napisan.

Dekadski broj u binaran pretvara se tako da se dekadski broj rastavi na faktore koji su potencije broja dva. Postupak se provodi na sljedeći način: prvo se nađe najviša potencija broja 2 koja se nalazi u dekadskome broju. Zatim se traži koja sljedeća niža potencija broja dva predstavlja ostatak i tako sve dok se ne dođe do najniže potencije bez ostatka.

3. Primjer: Pretvorba dekadskoga broja 55 u binaran.

$$55_{10} = 32 + 16 + 8 + 4 + 2 + 1 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 = 110111_2$$

55 – 32 = 23
23 – 16 = 7
8 ne postoji
7 – 4 = 3
3 – 2 = 1
1 – 1 = 0

4. Primjer: Pretvorba dekadskoga broja 59 u binaran

59 : 2 = 29 + ostatak 1
29 : 2 = 14 + ostatak 1
14 : 2 = 7 + ostatak 0
7 : 2 = 3 + ostatak 1
3 : 2 = 1 + ostatak 1
1 : 2 = 0 + ostatak 1

$59_{10} = 1\ 1\ 1\ 0\ 1\ 1_2$

Broj s razlomkom - razlomljen (*rational*), u dekadskome brojevnome sustavu pretvara se u binaran broj metodom uzastopnoga množenja dekadskoga broja brojem 2 (osnovica binarnoga brojevnoga sustava). Ako je rezultat množenja veći od jedan, znamenka binarnoga broja je 1, a ako je rezultat množenja manji od jedan, binarna znamenka je 0. Ako je rezultat množenja veći od jedan, mora se prije sljedećega množenja umanjiti za jedan.

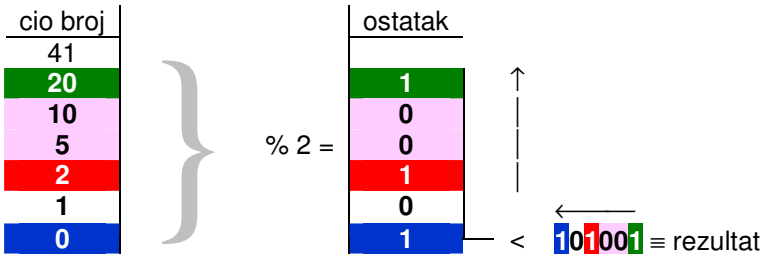
5. Primjer: Pretvorba dekadskoga broja 0,6285 u binaran.

$0{,}6875_{10} = 0{,}1\,0\,1\,1_2$

$0{,}6875 \times 2 = 1{,}375 = 0{,}375 + 1$ ostatak 1
$0{,}375 \times 2 = 0{,}75 = 0{,}75 + 0$ ostatak 0
$0{,}75 \times 2 = 1{,}5 = 0{,}5 + 1$ ostatak 1
$0{,}5 \times 2 = 1 = 0{,}0 + 1$ ostatak 1

6. Primjer: Pretvorba dekadskoga broja 41 u binaran.

Dekadski broj može se pretvoriti u binaran uzastopnim dijeljenjem dekadskoga broja brojem 2. Ako se pri dijeljenju dobije ostatak, znamenka binarnoga broja je 1, a ako ostatka nema, znamenka je 0. Prvo dijeljenje daje znamenku *najnižega* brojevnoga mjesta (dioba modulo 2):



Pretvorba dekadskih cijelih brojeva u bilo koji brojevni sustav baze *r*, slična je ovome primjeru, osim što se dijeli brojem *r* umjesto brojem 2 i bilježi se ostatak.

1.1.5. 1.1.5. OKTALNI BROJEVNI SUSTAV

Oktalni brojevni sustav ima osam osnovnih znamenaka: 0, 1, 2, 3, 4, 5, 6 i 7 pa mu je baza 8. Težine cjelobrojnih mjesta: $8^0 = 1$, $8^1 = 8$, $8^2 = 64$, $8^3 = 512$, …. Mjesta desno od zareza imaju težine 8^{-1}, 8^{-2}, 8^{-3} …. Najveći broj što ga se može napisati s *n* znamenaka je $8^n–1$. Opći prikaz broja u oktalnome brojevnome sustavu:

$$O = o_m \times 8^m + o_{m-1} \times 8^{m-1} + \ldots + o_2 \times 8^2 + o_1 \times 8^1 + o_0 \times 8^0 + o_{-1} \times 8^{-1} + o_{-2} \times 8^{-2} + \ldots + o_{-(n-1)} \times 8^{-(n-1)} + o_{-n} \times 8^{-n}$$

Koeficijenti *o* označavaju znamenke oktalnoga brojevnoga sustava.

7. Primjer: Opći prikaz broja u oktalnome brojevnome sustavu

8^2 64
8^1 8
8^0 1 … težina mjesta
8^{-1} 0,125
8^{-2} 0,015625

$234{,}56 = 2 \times 8^2 + 3 \times 8^1 + 4 \times 9^0 + 5 \times 8^{-1} + 6 \times 8^{-2}$

… znamenka najnižega mjesta - najmanje težine LSD (*least significant digit*)
… ~~dekadska~~ točka, zarez (~~*decimal*~~ *point*)
… znamenka najvišega mjesta - najveće težine MSD (*most significant digit*)

1.1.6. 1.1.6. PRETVORBA BROJEVA IZMEĐU OKTALNOG I DRUGIH BROJEVNIH SUSTAVA

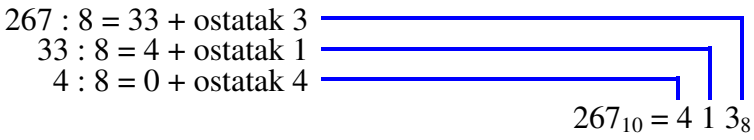
Pretvorba brojeva oktalnoga sustava u odgovarajući dekadski broj provodi se istim postupkom kao i binarnoga u dekadski. Razlika je samo u težinama brojevnih mjesta. Svaka znamenka oktalnoga sustava množi se svojom težinom, a dobiveni iznosi se zbroje.

8. Primjer: Pretvorba oktalnoga broja 237,51 u dekadski.

$$237{,}51_8 = 2 \times 8^2 + 3 \times 8^1 + 7 \times 8^0 + 5 \times 8^{-1} + 1 \times 8^{-2} =$$
$$= 2 \times 64 + 3 \times 8 + 7 \times 1 + 5 \times 0{,}125 + 1 \times 0{,}015625 = 159{,}640625_{10}$$

Pretvorba dekadskoga broja u odgovarajući oktalni može se provesti istim postupcima kao i kod pretvorbe u binaran broj. Ako je to metoda uzastopnoga dijeljenja dekadskoga broja bazom oktalnoga brojevnoga sustava, dobiveni ostaci dijeljenja označavaju znamenke oktalnoga brojevnoga sustava.

9. Primjer: Pretvorba dekadskoga broja 267 u oktalni.



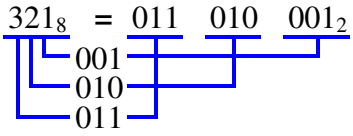
1.1.6.1. 1.1.6.1. Pretvorba oktalnoga broja u binaran broj - koder (8, 3)

Broj u oktalnome brojevnome sustavu pretvara se u broj binarnoga sustava tako da se svaka oktalna znamenka nadomjesti odgovarajućim binarnim brojem od 3 bita.

Tablica 1.2. Znamenke oktalnoga sustava i njihovi binarni ekvivalenti

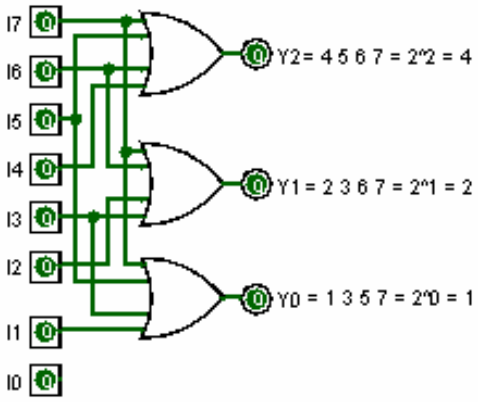
Oktalna znamenka	0	1	2	3	4	5	6	7
Binaran ekvivalent	000	001	010	011	100	101	110	111

10. Primjer: Pretvorba oktalnoga broja 321 u binaran



I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0	
0	0	0	0	0	0	0	1	0	0	0	I_0
0	0	0	0	0	0	1	0	0	0	1	I_1
0	0	0	0	0	1	0	0	0	1	0	I_2
0	0	0	0	1	0	0	0	0	1	1	I_3
0	0	0	1	0	0	0	0	1	0	0	I_4
0	0	1	0	0	0	0	0	1	0	1	I_5
0	1	0	0	0	0	0	0	1	1	0	I_6
1	0	0	0	0	0	0	0	1	1	1	I_7

$$Y_0 = I_1 + I_3 + I_5 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_2 = I_4 + I_5 + I_6 + I_7$$



1.1.6.2. 1.1.6.2. Pretvorba binarnoga broja u oktalni broj - dekoder (3, 8)

Broj iz binarnoga brojevnoga sustava pretvara se u oktalni tako da se binaran broj razdijeli u skupine od po 3 binarne znamenke počevši od najnižega brojevnoga mjesta. Svaka skupina binarnih znamenaka predočava se ekvivalentnom oktalnom znamenkom. Dobivene oktalne znamenke tvore oktalni broj.

11. Primjer: Pretvorba binarnoga broja 1011101010 u oktalni.

$$1011101010_2 = 001\ 011\ 101\ 010 = 1352_8$$
$$1\quad 3\quad 5\quad 2$$

1.1.7. 1.1.7. HEKSADEKADSKI BROJEVNI SUSTAV

Baza heksadekadskoga brojevnoga sustava je 16 pa sustav ima 16 znamenaka. Za opisati znamenke od nula do devet koriste se isti znakovi kao i za dekadski brojevni sustav. Zato što za znamenke od 10_{16} do 15_{16} ne postoje posebni simboli, za iskazati te brojeve koristi se šest slova abecede: A, B, C, D, E i F.

Tablica 1.3. Znamenke heksadekadskoga brojevnoga sustava

Heksadekadska znamenka	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dekadski broj	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Heksadekadski brojevni sustav najčešće se koristi za prikaz memorijskoga stanja procesorom upravljanih uređaja zbog jednostavne pretvorbe iz binarnoga brojevnoga sustava i obrnuto.

1.1.8. 1.1.8. PRETVORBA BROJEVA IZMEĐU HEKSADEKADSKOGA I DRUGIH BROJEVNIH SUSTAVA

Brojevi heksadekadskoga brojevnoga sustava pretvaraju se u dekadske brojeve tako da se vrijednost svake znamenke pomnoži težinom brojevnoga mjesta, a dobiveni iznosi se zbroje.

12. Primjer: Pretvorba između heksadekadskoga i ostalih brojevnih sustava

16^3
16^2
16^1
16^0 … težine mjesta

$6A5C = 6 \times 16^3 + A \times 16^2 + 5 \times 16^1 + C \times 16^0 = 35134_{10} = 0110\ 1010\ 0101\ 1100_2 = 65134_8$

… znamenka najnižega mjesta - najmanje težine LSD (*least significant digit*)
… znamenka najvišega mjesta - najveće težine MSD (*most significant digit*)

13. Primjer: Pretvorba heksadekadskoga broja 2D7A u dekadski.

$$2D7A_{16} = 2\times16^3 + D\times16^2 + 7\times16^1 + A\times16^0 = 2\times4096 + 13\times256 + 7\times16 + 10\times1 = 11642_{10}$$

Pretvorba dekadskoga broja u heksadekadski može se provesti istim postupkom kao i u ostale spomenute brojevne sustave, uzastopnim dijeljenjem dekadskoga broja sa 16, tj. bazom sustava. Ostatak dijeljenja označava heksadekadske znamenke.

14. Primjer: Pretvorba dekadskoga broja 235 u heksadekadski.

235 : 16 = 14 + ostatak 11
14 : 16 = 0 + ostatak 14

$235_{10} = E\ B_{16}$

Broj u heksadekadskome sustavu pretvara se u binaran broj tako da se svaka znamenka heksadekadskoga brojevnoga sustava nadomjesti odgovarajućim binarnim brojem, tj. kombinacijom od 4 bita prema tablici 1.4.

Tablica 1.4. Znamenke heksadekadskoga sustava i njihovi binarni ekvivalenti

Heksadekadska znamenka	Binaran ekvivalent	Heksadekadska znamenka	Binaran ekvivalent
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

15. Primjer: Pretvorba heksadekadskoga broja B7C u binaran.

$$B7C_{16} = 1011\ 0111\ 1100_2$$

Za pretvorbu binarnoga broja u heksadekadski, potrebno je binaran broj razdijeliti u skupine od četiri binarne znamenke počevši od najnižega mjesta. Svaka skupina binarnih znamenaka predočava se ekvivalentnom heksadekadskom znamenkom prema tablici 1.4.

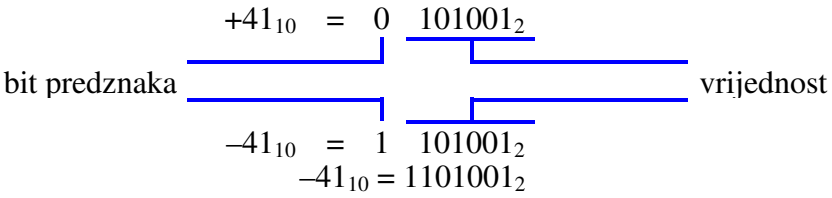
16. Primjer: Pretvorba binarnoga broja 101101001001 u heksadekadski.

$$101101001001_2 = 1011\ 0100\ 1001 = B49_{16}$$

1.1.9. 1.1.9. PRIKAZ CIJELIH BROJEVA

Za prikazivanje pozitivnih i negativnih brojeva ili brojeva s predznakom (*signed numbers*) koristi se dodatan bīt za predznak (*sign bit*). Bīt oznake *pozitivnoga* broja je 0, a *negativnoga* broja je 1. Ostali bitovi čine iznos, veličinu ili vrijednost (*magnitude*) broja. Iznosi negativnih brojeva mogu se prikazati na više načina. Jedna od mogućnosti za prikaz iznosa negativnih brojeva jest korištenje iznosa za pozitivne brojeve (*true-magnitude form*). Pozitivni i negativni brojevi razlikuju se samo u bītu predznaka.

17. Primjer: Prikaz brojeva +41 i -41 pomoću bita za predznak i binarnoga broja.

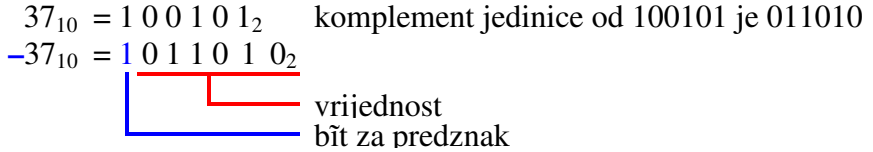


Drugi način prikaza iznosa negativnoga broja jest korištenje komplementa[2]. Moguće je koristiti:

- komplement baze (*radix*) '2' ili komplement do baze binarnoga sustava (2'*s-complement*) ili
- komplement umanjene baze (*diminished radix*) '1' ili komplement do najvećega osnovnoga broja (1'*s-complement*).

U binarnome brojevnome sustavu komplement jedinice nekoga broja dobije se zamjenom nula u jedinice i jedinica u nule.

18. Primjer: Prikaz broja -37 pomoću bita za predznak i komplementa jedinice



1.1.9.1. Pretvorba dekadskoga broja u broj, bilo kojega brojevnoga sustava!

19. Pretvorba dekadskoga broja u broj, bilo kojega brojevnoga sustava!

```
//#include <conio.h> //_getch();
```

```
#include <iostream>
using namespace std;
void main(){
int broj,duzina,baza;
char polje[18];//umjesto običnoga polja,
koristiti vector
//Problem veličine polja[] odražava se tek
//pri izlasku iz programa nakon 2. Ctrl+Z
cout<<"Prekid upisa: Ctrl+Z\n";
do{
broj=0;
cout<<"Upisi cio dekadski broj: ";
cin>>broj;
//duzina=strlen(char* broj);
cout<<"Upisi bazu sustava: ";
cin>>baza;
//cout<<" "<<broj<<endl;
if(broj==baza)cout<<"Uz broj==baza, rezultat
je: ";
cout<<itoa(broj,polje,baza)<<endl;
}while(getchar()!=EOF);
//putchar(c);
}
```

```
C:\windows\system32\cmd.exe
Upisi dekadski broj: 5
Upisi bazu sustava: 5
Uz broj==baza, rezultat je: 10
Upisi dekadski broj: 4
Upisi bazu sustava: 4
Uz broj==baza, rezultat je: 10
Upisi dekadski broj: 3
Upisi bazu sustava: 3
Uz broj==baza, rezultat je: 10
Upisi dekadski broj: 2
Upisi bazu sustava: 2
Uz broj==baza, rezultat je: 10
Upisi dekadski broj: ^Z
Upisi bazu sustava: 0
^Z
Press any key to continue . . .
```

[2] *komplement* … nadopuna

1.1.10. OBIČAN KOMPLEMENT

Komplementi se koriste u digitalnim uređajima za pojednostavniti operaciju oduzimanja i za rukovanje logičkim operacijama.

Pojednostavnjenjem, dobiju se jeftiniji sklopovi za provedbu operacija. Postoje dvije vrste komplementa za svaki sustav baze *r*: komplement i umanjen (*diminish*) komplement baze. Prvi se naziva *r* komplement, a drugi (*r*–1) komplement. Ako se vrijednost baze *r* zamijeni imenom, dobiju se dvije vrste za svaki brojevni sustav:

- Za binaran sustav (baza 2), komplement su: komplement dvojke (2'*s complement*) i komplement jedinice (1'*s complement*).
- Za dekadski sustav (baza 10), komplementi su: komplement desetke (10'*s complement*) i komplement devetke (9'*s complement*).
- Za oktalni sustav (baza 8), komplementi su: komplement osmice (8'*s complement*) i komplement sedmice (7'*s complement*).
- Za heksadekadski sustav (baza 16), komplementi su: komplement šesnaestice (16'*s complement*) i komplement petnaestice (15'*s complement*).

Običan komplement binarnoga broja (*komplement jedinice*) dobije se zamjenom svih jedinica nulama i obratno. Iznos broja koji se dobije na ovaj način računa se prema izrazu:

$$\overline{N}_2 = 2^n - 1 - N_2$$

Običan komplement nije pogodan za izražavanje prirodnih brojeva, jer nije jednoznačno određena vrijednost nule, naime običan komplement od 0000 iznosi 1111, ali služi za jednostavno računanje punoga komplementa, jer vrijedi:

$$N_2^{'} = \overline{N}_2 + 1$$

Ako se $\overline{N}_2$ u ovoj jednadžbi zamijeni prethodnom jednadžbom, dobije se:

$$N_2^{'} = 2^n - 1 + 1 - N_2 \Rightarrow N_2^{'} = 2^n - N_2$$

ili za prethodan primjer: 1111 + 1 = 10000.

Komplement baze nekoga broja u nekome brojevnome sustavu dobije se tako da se znamenke zamijene razlikama do broja baze (2, 8, 10, 16) pa im se pribroji 1, tj. *komplementu umanjene baze* doda se 1.

Komplement devetke dekadskoga broja: 546700 je 999999 – 546700 = 453299.

Komplement desetke dekadskoga broja: 546700 je 453299 + 1 = 453300.

Komplement devetke dekadskoga broja: 012398 je 999999 – 012398 = 987601.

Komplement desetke dekadskoga broja: 012398 je 012398 + 1 = 987699.

Komplement dvojke nekoga binarnoga broja dobije se tako da se jedinice zamijene nulama, a nule jedinicama te se pribroji 1, tj. *komplementu jedinice* doda se 1.

20. Primjer: Prikaz broja -41 pomoću bita za predznak i komplementa dvojke

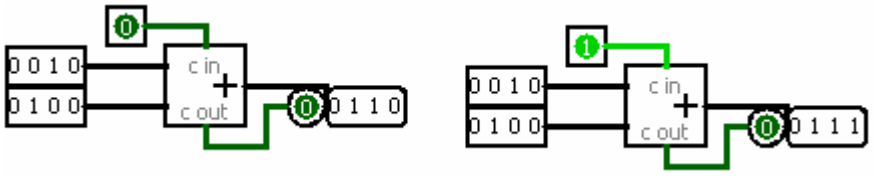
37_{10} = 1 0 1 0 0 1_2 komplement dvojke od 100101 je 011010 = 011011

-37_{10} = 1 1 0 1 0 0 1_2

vrijednost

bit za predznak

Prikaz komplementa dvojke simulacijskim krugom LogiSim dan je u nastavku.



1.1.11. 1.1.11. KOMPLEMENTI BINARNIH BROJEVA

Budući da je baza sustava (*radix*) za binaran broj 10_2, smanjena baza je 1_2. Komplement smanjene baze nađe se oduzimanjem binarnoga broja od 1_2. To se obično zove *komplement jedinice*. Na primjer, komplement jedinice za 0 nađe se kao 1–0 = 1, a komplement jedinice za 1 nađe se kao 1–1 = 0. U biti, komplement jedinice dobiva se jednostavnim obrtanjem (ili "zrcaljenjem") svakoga bita u binarnome broju pa je za 100101_2 komplement jedinice jednak 011010_2. Komplement baze (ili "komplement dvojke") binarnoga broja nađe se tako da se prvo izračuna komplement jedinice, a zatim se broj 1 pribroji tome broju. Komplement jedinice za 101101_2 je 010010_2, a komplementa dvojke za 010010_2+1_2, ili 010011_2.

1.1.11.1. Komplementi s predznakom

U krugovima koji koriste binarnu matematiku, oblikovatelj kruga može se odlučiti za korištenje komplementa za negativne brojeve i odrediti najznačajniji bit kao bit predznaka. Ako je tako, ostali bitovi su vrijednost broja. Ako je najznačajniji bit 1, on označava negativne brojeve, a ako je najznačajniji bit 0, tada je broj pozitivan, a vrijednost broja određuje se uzimanjem komplementa preostalih bitova. Ali ako je najznačajniji bit 1, onda je broj negativan, a vrijednost broja određuje se uzimanjem komplementa jedinice broja. Tako je: 0111 = +7, a 1000 = –7 (komplement jedinice za 1000 je 0111). Sljedeća tablica može pomoći u bistrenju ovoga koncepta:

Tablica 1.5. Komplement jedinice negativnih brojeva

dekadski	pozitivni	negativni
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000

Binarnim brojem s 4 bita, može se predstaviti bilo koji dekadski broj od –7 do +7. Ali, uočite da poput sustava predznak-i-vrijednost, postoje dvije vrijednosti za 0, jedna pozitivna i jedna negativna. To zahtijeva dodatne sklopove za testiranje obje vrijednosti za 0 nakon operacije poput oduzimanja. Kako bi se pojednostavnilo oblikovanje sklopa, možemo se odlučiti za korištenje *komplementa dvojke* negativnih brojeva te odrediti najznačajniji bit kao bit predznaka. U tome slučaju, ostali bitovi predstavljaju vrijednost broja.

Pri korištenju komplementa dvojke, ako je najznačajniji bit 0, onda je broj pozitivan i vrijednost broja određuje se iz preostalih bitova. Ako je najznačajniji bit 1, onda je broj negativan, a vrijednost broja određuje se uzimanje komplementa svakoga od bitova, a zatim se pribraja 1 (izrada komplementa dvojke). Tako je: 0111 = 7, a 1001 = –7 (komplement jedinice od 1001 je 0110, a komplement dvojke od 0110 je 0110+1 = 0111). Sljedeća tablica može pomoći u razjašnjavanju ovoga koncepta:

Tablica 1.6. Komplement dvojke negativnih brojeva

dekadski	pozitivni	negativni
0	0000	10000
1	0001	1111
2	0010	1110
3	0011	1101
4	0100	1100
5	0101	1011
6	0110	1010
7	0111	1001

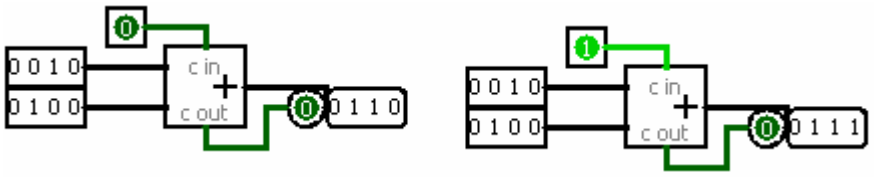
Komplement dvojke uklanja hirovitost označavanja nule na dva načina. Gornja tablica pokazuje da je nula ili 0000 ili 10000; a budući da se pretpostavlja rad s 4 bita, "1" se inicijalno odbacuje, ostavljajući 4 nule (0000) bez obzira je li nula pozitivan ili negativan broj.

1.1.11.2. Izračun komplementa dvojke

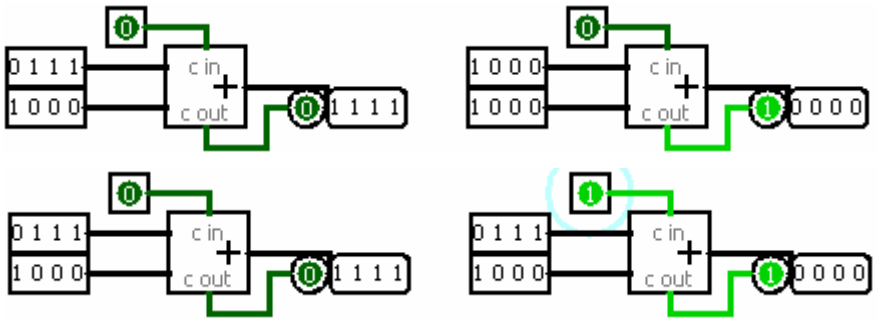
Komplement dvojke - korijen (*radix*) računa se pomoću komplementa jedinice broja i nakon toga dodavanjem jedinice. Za strojeve, to je najučinkovitiji način izračuna. Postoji način koji je puno lakši za ljude kada koriste izračun komplementa dvojke nekoga broja. Započne se najmanje značajnim bitom (bit sasvim desno), a zatim se čita broj s desna u lijevo. Potraži se prva "1", a zatim preokrene svaki bit lijevo od te "1".

Zbrajalo je napravljeno tako da prvi broj koji se zbraja ulazi na ulaz lijevo gore, a drugi broj ulazi na ulaz lijevo dolje.

Utjecaj i ponašanje bita posudbe (*Carry in*)



Utjecaj i ponašanje bita pretek-prijenos (*Carry out*)



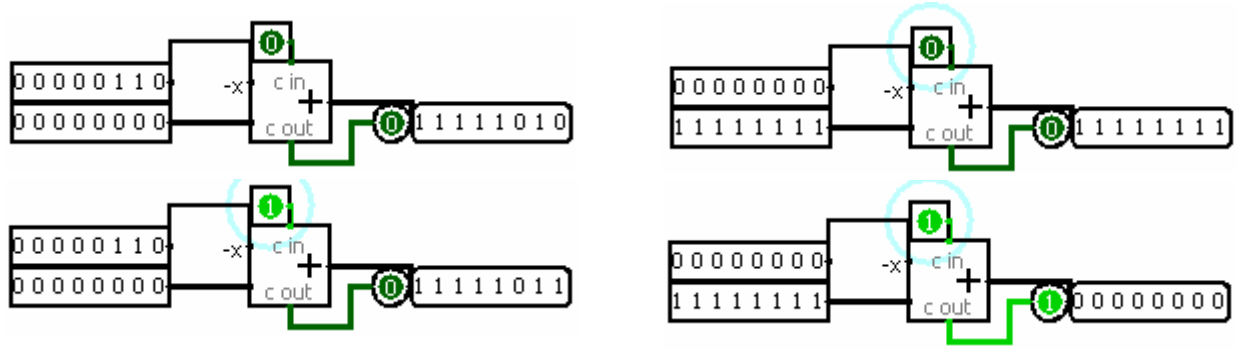
Ulaz "*bita za posudbu*" (***carry in bit***) je na vrhu (on se zbraja s bitom postavljenim sasvim desno na izlazu). Primjena je u *punome komplementu* ili *komplementu dvojke*, a koristi se za označavanje ***negativnih*** brojeva. *Pun komplement* izračuna se tako da se najprije izračuna *običan komplement* te mu se doda jedinica.

komplement dvojke broja 6:

```
00000110  +6
11111001  običan komplement od +6
      + 1 dodaj 1
11111010  -6 u komplementu dvojke
```

pun komplement od 0 iznosi:

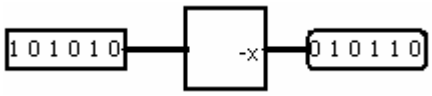
```
00000000  0
11111111  običan komplement od 0
      + 1 dodaj 1
100000000 -0 u komplementu dvojke
```



U navedenome primjeru *punoga komplementa* od 0, jedinica predstavlja deveti bit, te se stoga u sustavu od 8 bitova odbacuje pa rezultat opet predstavlja nulu. *Komplement dvojke* omogućuje jednoznačno određivanje nule i zbog toga je izabran za predstavljanje cijelih brojeva, a ne običan komplement (komplement jedinice).

U LogiSimu, bît, što se kao rezultat zbrajanja prenosi i naznačuje da je potrebno osigurati još jedno binarno mjesto - npr. u registru (***carry out bit***), izlazi na dnu. Zbroj dvaju brojeva pojavljuje se u čvoru na desnoj strani.

Kao primjer, komplement dvojke broja 101010 oblikuje se počevši od najmanje značajnoga bita (0 na desnoj strani) i obrađuju se bitovi prema lijevoj strani, u potrazi za prvom 1, koja je u ovome primjeru na drugome mjestu s desne strane.



Svaki bit u lijevome stupcu obrće se i ukupan vektor završava kao 010110.

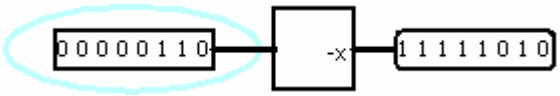
Evo još nekoliko primjera:

Tablica 1.7. Komplementi dvojke

Broj	LogiSim prikaz	Komplement dvojke
0110100	0 1 1 0 1 0 0 -x 1 0 0 1 1 0 0	1001100
11010	1 1 0 1 0 -x 0 0 1 1 0	00110
001010	0 0 1 0 1 0 -x 1 1 0 1 1 0	110110
1001011	1 0 0 1 0 1 1 -x 0 1 1 0 1 0 1	0110101
111010111	1 1 1 0 1 0 1 1 1 -x 0 0 0 1 0 1 0 0 1	000101001

21. Primjer: Računanje komplementa dvojke ulaznoga broja

Logisim uključuje uređaj koji računa komplement dvojke ulaznoga broja. Zove se "negator" i nalazi se u mapi "Aritmetika". On ima oznaku "-x" na svome istočnome dijelu.



1.1.12. 1.1.12. PREGLED KLJUČNIH POJMOVA

binarna znamenka (*binary digit-bit*)

- znamenka binarnoga brojevnoga sustava

binarna znamenka najnižega brojevnoga mjesta LSB (*least significant bit*)

- binarna znamenka najmanje vrijednosti brojevnoga mjesta, krajnja desna znamenka binarnoga broja

binarna znamenka najvišega brojevnoga mjesta MSB (*most significant bit*)

- binarna znamenka najveće vrijednosti brojevnoga mjesta, krajnja lijeva znamenka binarnoga broja

binaran brojevni sustav (*binary number system*)

- brojevni sustav s dvije znamenke: 0 i 1

binaran zarez (*binary point*)

- zarez koji razdvaja cjelobrojna od razlomljenih mjesta u binarnom brojevnom sustavu

brojno mjesto

- položaj znamenke u bilo kojemu broju

dekadski brojevni sustav (*decimal number system*)

- brojevni sustav s deset znamenaka

digitalna elektronika (*digital electronics*)

- područje elektronike u kojemu signali mogu imati dva iznosa kojima se pridružuju znamenke 0 i 1 što omogućuje prikaz podataka u brojčanome obliku binarnoga brojevnoga sustava

komplement dvojke (2*'s-complement*)

- ili komplement do baze, dobije se tako da se komplementu jedinice doda jedinica

heksadekadski brojevni sustav (*hexadecimal number system*)

- brojevni sustav sa 16 znamenaka

oktalni brojevni sustav (*octal number system*)

- brojevni sustav s 8 znamenaka

komplement jedinice (1*'s-complement*)

- ili komplement do najvećega broja, u binarnome brojevnome sustavu dobije se međusobnom zamjenom nula i jedinica.

težina brojevnoga mjesta

- vrijednost brojevnoga mjesta, može se prikazati kao potencija osnovice (baze) brojevnoga sustava

znamenka najnižega brojevnoga mjesta (*least significant digit*)

- znamenka najmanje vrijednosti (težine) brojevnoga mjesta, krajnja desna znamenka broja bilo kojega brojevnoga sustava

znamenka najnižega brojevnoga mjesta LSD (*least significant digit*)

- znamenka najmanje vrijednosti (težine) brojevnoga mjesta, krajnja desna znamenka broja bilo kojega brojevnoga sustava

znamenka najvišega brojevnoga mjesta MSD (*most significant digit*)

- znamenka najviše vrijednosti (težine) brojevnoga mjesta, krajnja lijeva znamenka broja bilo kojega brojevnoga sustava

Tablica 1.8. Pregled brojevnih sustava

Brojevni sustav	Dekadski	Binarni	Oktalni	Heksadekadski
Znamenke	0	0	0	0
	1	1	1	1
	2		2	2
	3		3	3
	4		4	4
	5		5	5
	6		6	6
	7		7	7
	8			8
	9			9
				A
				B
				C
				D
				E
				F
Baza sustava	10	2	8	16
	$10^0 = 1$	$2^0 = 1$	$8^0 = 1$	$16^0 = 1$
	$10^1 = 10$	$2^1 = 2$	$8^1 = 8$	$16^1 = 16$
	$10^2 = 100$	$2^2 = 4$	$8^2 = 64$	$16^2 = 256$

Brojevni sustav	Dekadski	Binarni	Oktalni	Heksadekadski
	$10^3 = 1000$	$2^3 = 8$	$8^3 = 512$	$16^3 = 4096$
Prvih 16 brojeva	0	0	0	0
Težine mjesta	1	1	1	1
	2	10	2	2
	3	11	3	3
	4	100	4	4
	5	101	5	5
	6	110	6	6
	7	111	7	7
	8	1000	10	8
	9	1001	11	9
	10	1010	12	A
	11	1011	13	B
	12	1100	14	C
	13	1101	15	D
	14	1110	16	E
	15	1111	17	F
	16	10000	20	10

Tablica 1.9. Prikaz cijelih brojeva

Dekadski broj	Prikaz pomoću binarnoga broja	Prikaz pomoću komplementa jedinice	Prikaz pomoću komplementa dvojke
	$P2^32^22^12^0$	$P2^32^22^12^0$	$P2^32^22^12^0$
15	0 1 1 1 1	0 1 1 1 1	0 1 1 1 1
14	0 1 1 1 0	0 1 1 1 0	0 1 1 1 0
13	0 1 1 0 1	0 1 1 0 1	0 1 1 0 1
12	0 1 1 0 0	0 1 1 0 0	0 1 1 0 0
11	0 1 0 1 1	0 1 0 1 1	0 1 0 1 1
10	0 1 0 1 0	0 1 0 1 0	0 1 0 1 0
9	0 1 0 0 1	0 1 0 0 1	0 1 0 0 1
8	0 1 0 0 0	0 1 0 0 0	0 1 0 0 0
7	0 0 1 1 1	0 0 1 1 1	0 0 1 1 1
6	0 0 1 1 0	0 0 1 1 0	0 0 1 1 0
5	0 0 1 0 1	0 0 1 0 1	0 0 1 0 1
4	0 0 1 0 0	0 0 1 0 0	0 0 1 0 0
3	0 0 0 1 1	0 0 0 1 1	0 0 0 1 1
2	0 0 0 1 0	0 0 0 1 0	0 0 0 1 0
1	0 0 0 0 1	0 0 0 0 1	0 0 0 0 1
0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
–1	1 0 0 0 1	1 1 1 1 0	1 1 1 1 1
–2	1 0 0 1 0	1 1 1 0 1	1 1 1 1 0
–3	1 0 0 1 1	1 1 1 0 0	1 1 1 0 1
–4	1 0 1 0 0	1 1 0 1 1	1 1 1 0 0
–5	1 0 1 0 1	1 1 0 1 0	1 1 0 1 1
–6	1 0 1 1 0	1 1 0 0 1	1 1 0 1 0
–7	1 0 1 1 1	1 1 0 0 0	1 1 0 0 1
–8	1 1 0 0 0	1 0 1 1 1	1 1 0 0 0
–9	1 1 0 0 1	1 0 1 1 0	1 0 1 1 1
–10	1 1 0 1 0	1 0 1 0 1	1 0 1 1 0
–11	1 1 0 1 1	1 0 1 0 0	1 0 1 0 1
–12	1 1 1 0 0	1 0 0 1 1	1 0 1 0 0
–13	1 1 1 0 1	1 0 0 1 0	1 0 0 1 1
–14	1 1 1 1 0	1 0 0 0 1	1 0 0 1 0
–15	1 1 1 1 1	1 0 0 0 0	1 0 0 0 1

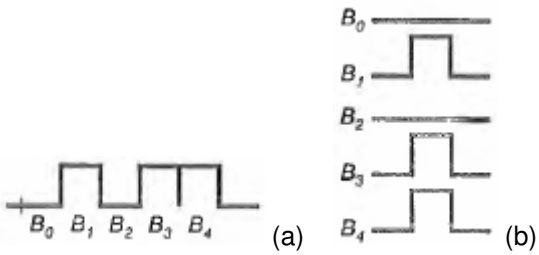
1.1.13. 1.1.13. PITANJA I ZADACI ZA PONAVLJANJE

1. Objasnite razlike između dekadskoga i binarnoga brojevnoga sustava.
2. Koji se najveći broj (izražen dekadsko) može napisati s 8 znamenaka binarnoga sustava?
   255

3. Koliko je binarnih znamenaka potrebno za prikaz dekadskoga broja 47?

   6

4. Prikažite binaran signal 11010 u serijskome i paralelnome obliku.



5. Pretvorite binaran broj 110110 u dekadski.

   54

6. Pretvorite dekadski broj 53 u binaran.

   110101

7. Navedite značajke oktalnoga brojevnoga sustava.

8. Koji se najveći broj (izražen dekadski) može napisati s četiri znamenke oktalnoga sustava?

   4095

9. Pretvorite oktalni broj 465 u dekadski.

   309

10. Pretvorite dekadski broj 372 u oktalni.

    564

11. Pretvorite oktalni broj 524 u binaran.

    101010100

12. Pretvorite binaran broj 11001110 u oktalni.

    316

13. Navedite značajke heksadekadskoga brojevnoga sustava,

14. Koji se najveći broj (izražen dekadski) može napisati s tri znamenke heksadekadskoga sustava?

    4095

15. Koje su težine mjesta znamenaka najnižega i najvišega brojevnoga mjesta heksadekadskoga broja A3F,5D?

    16^{-2} i 16^{2}

16. Pretvorite heksadekadski broj 12BF u dekadski.

    4799

17. Pretvorite dekadski broj 3127 u heksadekadski.

    C37

18. Pretvorite heksadekadski broj 2C4E u binaran.

    10110001001110

19. Pretvorite binaran broj 100101000111111 u heksadekadski.

    4A3F

20. Pretvorite heksadekadski broj D3A u oktalni.

    6472

21. Pretvorite oktalni broj 5437 u heksadekadski.

    B1F

22. Prikažite broj –25 pomoću binarnoga broja, komplementa jedinice i komplementa dvojke.

    111001, 100110, 100111

1.2. 1.2. KÔDOVI

(3.5.1.1. Morseov kod), (3.5.1.2. Brailleovom pismu)

Podaci iz digitalnih uređaja preslikavaju se u binarne znamenke koristeći standardizirane kodove. Oni omogućuju da se osim brojeva mogu prikazivati slova i znakovi. Kôd je definirana kombinacija binarnih znamenaka koja se dodjeljuje dekadskoj znamenci, slovu ili znaku.

Ako se kôdiranjem želi prikazati znamenke dekadskoga brojevnoga sustava, potrebno je koristiti kombinacije od najmanje četiri bita. S četiri bita može se dobiti $2^4 = 16$ različitih kombinacija. Kako je za prikaz znamenaka dekadskoga brojevnoga sustava potrebno svega 10 kombinacija, osmišljeni su brojni su načini za kôdiranje dekadskih znamenaka. Najčešći kodovi su: BCD, Excess-3, Aikenov i Grayev kôd.

Kodovi što osim brojeva omogućuju prikaz (kôdiranje) slova i znakova, nazivaju se alfanumerički kôdovi. To su kôdovi s više od četiri bita kako bi se postigao potreban broj kombinacija.

1.2.1. 1.2.1. BCD KÔD

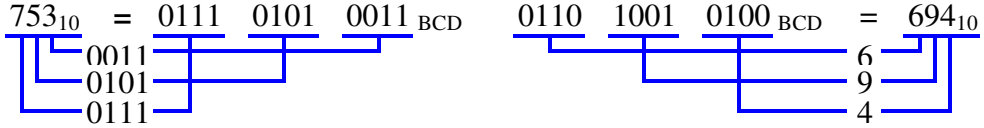
Za kôdiranje dekadskih znamenaka u BCD kôdu (*Binary Coded Decimal*) koristi se prvih deset kombinacija prirodnoga binarnoga niza od 4 bita. To znači da se svaka dekadska znamenka prikazuje pripadnim binarnim brojem. Stoga se ovaj kôd ponekad naziva i prirodni binarno-dekadski kôd ili kraće NBCD kôd (*Natural Binary Coded Decimal*).

BCD kôd naziva se težinski kôd (*weighted code*), jer bitovi kombinacija imaju težine 2^3, 2^2, 2^1 i 2^0. Zbroj težina brojevnih mjesta na kojima je binarna znamenka 1, daju vrijednost kôdirane dekadske znamenke. Kôd sadrži i kombinaciju 0000 što znači da se prekid u prijenosu podataka može shvatiti kao podatak 0. Zbog toga razloga, kombinacija binarnih znamenaka "sve 0" najčešće se dinamički i aktivno ne koristi, ali je dobrodošla za neke kratkotrajne i statičke provjere. Također, ta kombinacija u različitim dužinama neophodna je za dovođenje nekih sklopova u stanje početnih uvjeta.

Tablica 1.10. BCD kôd

Dekadska znamenka	Binarna 8421	
0	0000	Preostalih 6 binarnih kombinacija: 1010, 1011, 1100, 1101, 1110 i 1111, iako se izravno ne koriste, mogu poslužiti kao dodatna zalihost u dinamičkim uvjetima rada nekoga sustava ili za neka dodatna kodiranja
1	0001	
2	0010	
3	0011	
4	0100	
5	0101	
6	0110	
7	0111	
8	1000	
9	1001	

22. Primjer: Kodiranje i dekodiranje u BCD kôdu.



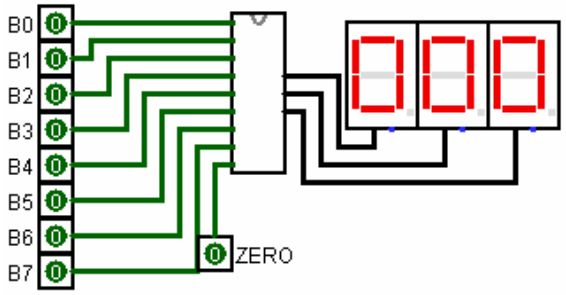
Potrebno je razlikovati broj prikazan u binarnome brojevnome sustavu od istoga broja prikazanoga u binarnome kôdu, iako se u oba slučaja radi o nizu bitova. Kombinacija bitova u binarnome brojevnome sustavu označava uvijek određen broj.

Kombinacija bitova u kôdu može označavati broj, ali i znakove ili slova, dakle općenito neki podatak.

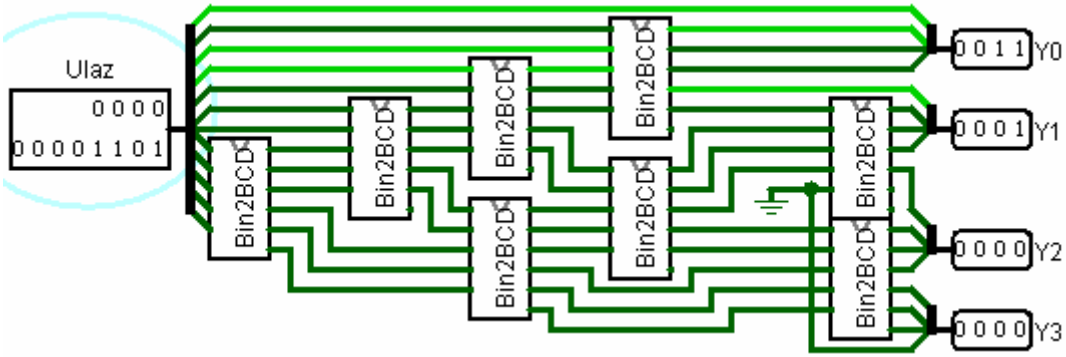
23. Primjer: Binarna kombinacija 10000110 u dekadskome i BCD kodu

U dekadskome sustavu, binarnoj kombinaciji 10000110_2 broj 134_{10}. Ista kombinacija u BCD kôdu odgovara dekadskome broju 86_{BCD}.

24. Primjer: Pretvorba binarnoga broja u BCD kôd pomoću simulacijskoga alata LogiSim 2.7.1.



25. Primjer: Sklop za pretvorbu binarnoga broja u BCD format



1.2.2. 1.2.2. EXCESS-3 KÔD

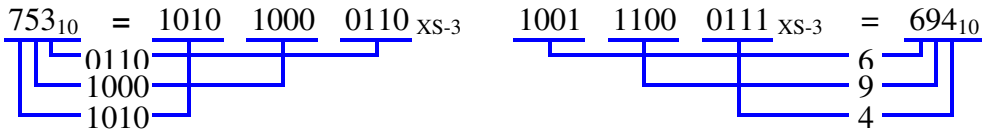
Za kôdiranje dekadskih znamenaka u Excess-3 kôdu (skraćeno XS-3 kôdu) koristi se srednjih deset kombinacija binarnoga niza od 4 bitova, a odbacuju se prve tri i zadnje tri kombinacije (vidi tablicu 1.11).

Tablica 1.11. Excess-3 kôd

Dekadska znamenka	Binarna kombinacija	Komplementi
0	0011	1100
1	0100	1011
2	0101	1010
3	0110	1001
4	0111	1000
5	1000	0111
6	1001	0110
7	1010	0101
8	1011	0100
9	1100	0011

Excess-3 kôd razlikuje se od BCD kôda i po tome što nije težinski, ali je samo-komplementan (*self-complemented*). To znači da se komplement bilo koje znamenke dobije zamjenom nula jedinicama i jedinica nulama. Osim toga, u Excess-3 kôdu ne pojavljuju se kombinacije sa sve četiri nule niti sve četiri jedinice, što može biti korisno za otkrivanje prekida u prijenosu podataka.

26. Primjer: Kodiranje i dekodiranje Excess-3 kôdom.



27. Primjer: Pretvorba BCD kôda u Excess-3 kôd

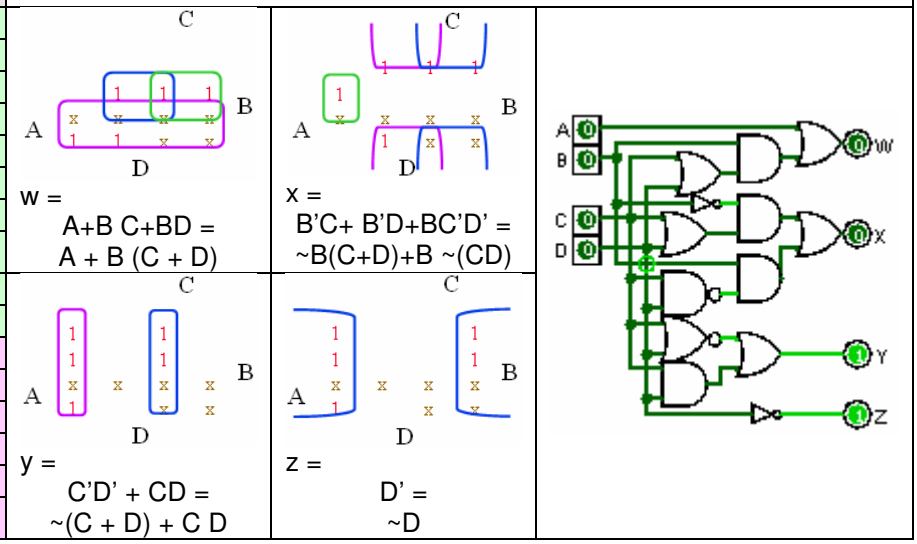
1.2.2.1. Algoritam:

1. odrediti broj ulaza i izlaza,
2. napisati tablicu istine,
3. minimizirati Booleov izraz za svaki izlaz i
4. generirati zahtijevani krug.

28. Primjer: Oblikovati krug za pretvorbu BCD kôda u Excess-3 kôd.

BCD kôd opisuje dekadske znamenke od 0 do 9 i sastoji se od 4 bita, a u ovome primjeru Excess-3 kôd koristi početne kombinacije binarnoga niza od 4 bita (16 kombinacija) i ne koristi zadnjih 6 kombinacija.

A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



1.2.3. 1.2.3. AIKENOV KÔD

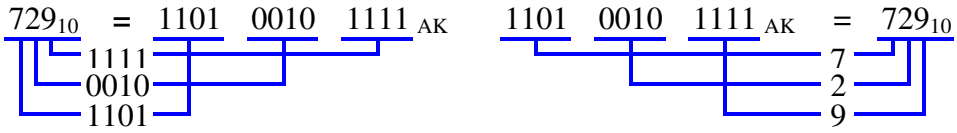
U Aikenovu[3] kôdu koristi se prvih pet i zadnjih pet kombinacija niza od 4 bita, a odbacuje se srednjih šest kombinacija. Kod je samo-komplementan i težinski, s težinama mjesta 2 4 2 1.

Tablica 1.12. Aikenov kôd

Dekadska znamenka	Binarna kombinacija 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

[3] Howard Aiken, američki znanstvenik sveučilišta Harvard, konstruktor prvoga elektromehaničkoga računala MARK I

29. Primjer: Pretvorba dekadskoga u Aikenov kôd



1.2.4. 1.2.4. GRAYEV KÔD

Grayev kôd nije težinski, a svaka kombinacija razlikuje od prethodne za samo jedan bit. Grayev kôd temelji se na zrcalnome (*reflected*) binarnome brojevnome sustavu. Stoga taj kôd spada u skupinu zrcalnih kôdova (*reflected codes*).

Brojevi zrcalnoga binarnoga brojevnoga sustava dobiju se na sljedeći način: znamenke 0 i 1 napišu se jedna ispod druge. Ispod njih se povuče zamišljena crta zrcaljenja (crta zrcaljenja), a ispod nje napišu se znamenke 1 i 0 kao zrcalna slika. Sada se ispred gornjih znamenaka dodaju nule, a ispred donjih jedinice. Na taj način dobije se skupina od 4 kombinacije po 2 bita.

0	00
1	01
—crta zrcaljenja	—
1	11
0	10

Ako se ispod njih povuče nova zrcalna crta (crta zrcaljenja) i ispod nje napišu zrcalne kombinacije pa se gornjima dodaju nule, a donjima jedinice, dobije se 8 kombinacija od 3 bita.

00	000
01	001
11	011
10	010
— crta zrcaljenja	——
10	110
11	111
01	101
00	100

Na isti način povuče se nova zrcalna crta (crta zrcaljenja) i ispod nje napišu zrcalne kombinacije pa se gornjima dodaju nule, a donjima jedinice te se dobije 16 kombinacija od 4 bita.

000	0000
001	0001
011	0011
010	0010
110	0110
111	0111
101	0101
100	0100
— crta zrcaljenja	———
100	1100
101	1101
111	1111
110	1110
010	1010
011	1011
001	1001
000	1000

Na ovaj način može se dobiti broj kombinacija po želji. Za Grayev kôd koristi se prvih deset kombinacija niza od 4 bita (tablica 1.13.).

Tablica 1.13. Zrcalni binarni brojevni sustav

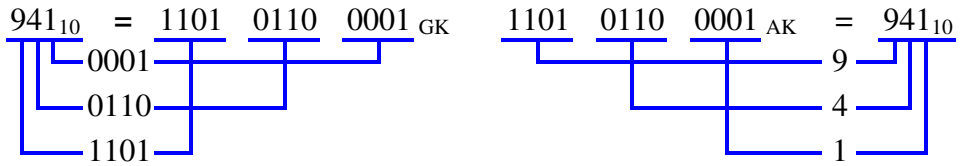
0	00
1	01
— crta zrcaljenja	—
1	11
0	10

00	000
01	001
11	011
10	010
— crta zrcaljenja	———
10	110
11	111
01	101
00	100

000	0000
001	0001
011	0011
010	0010
110	0110
111	0111
101	0101
100	0100
— crta zrcaljenja	———
100	1100
101	1101
111	1111
110	1110
010	1010
011	1011
001	1001
000	1000

Zrcalni binarni brojevni sustav	
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

30. Primjer: Kodiranje Grayevim kôdom.



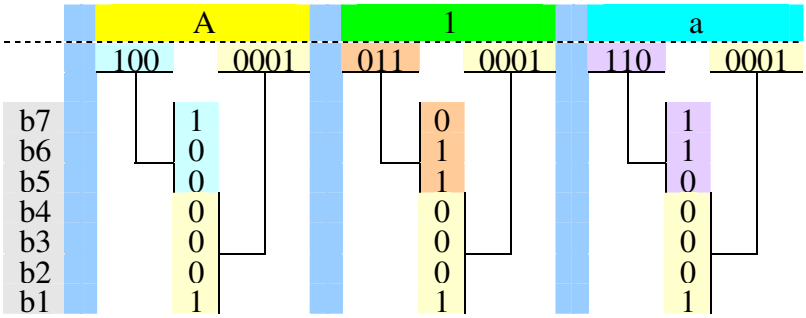
1.2.5. 1.2.5. ALFANUMERIČKI KÔDOVI

Među alfanumeričkim kôdovima najčešće je u uporabi kôd poznat pod nazivom ASCII (*American Standard Code for Information Interchange*). To je kôd od 7 bitova što daje 128 kombinacija. To je dovoljno za prikaz svih znamenaka, slova i znakova. Kôd se koristi u prijenosu podataka između računala i ulazno-izlaznih uređaja.

Tablica 1.14. ASCII kôd

					b7	0	0	0	0	1	1	1	1
					b6	0	0	1	1	0	0	1	1
					b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	¬	n	–
1	1	1	1	F	SI	US	/	?	O	_	O	DEL

31. Primjer: Kodiranje ASCII kôdom podatka A1a.

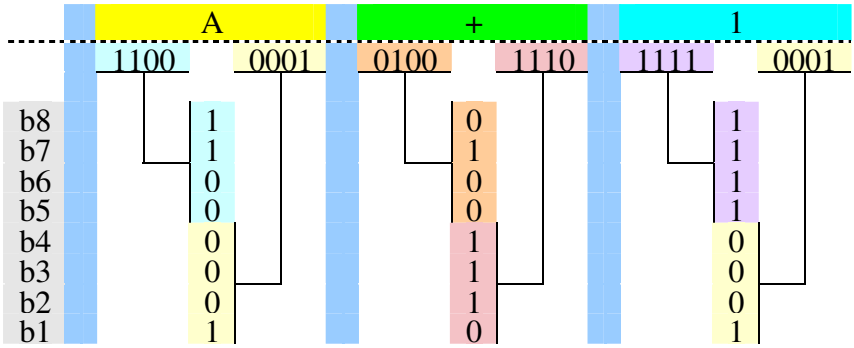


Uz ASCII kôd vrijedno je spomenuti i kôd od 8 bitova EBCDI (*Extended BCD Interchange Code*). Kôd daje ukupno 256 mogućih kôdnih kombinacija, što je više od potrebe za znamenke, slova i znakove pa ostaje velik dio kombinacija za uporabu kao upravljačke naredbe.

Tablica 1.15. EBCDI kôd

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	NUL				SP	&	_									0
0	0	0	1	1							/		a	j			A	J		1
0	0	1	0	2									b	k	s		B	K	S	2
0	0	1	1	3									c	l	t		C	L	T	3
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0	1	0	1	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0	1	1	0	6	LC	BS	EOB	UC					f	o	w		F	O	W	6
0	1	1	1	7	DEL	1L	PRE	EOT					g	p	x		G	P	X	7
1	0	0	0	8									h	q	y		H	Q	Y	8
1	0	0	1	9									i	r	z		I	R	Z	9
1	0	1	0	A		SM			¢	!	^	:								
1	0	1	1	B					.	$	,	#								
1	1	0	0	C					<	*	%	@								
1	1	0	1	D					(	)	·	'								
1	1	1	0	E					+	;	>	=								
1	1	1	1	F						¬	?	"								

32. Primjer: Kodiranje EBCDI kôdom podatka A + 1.



1.2.6. 1.2.6. KODOVI ZA OTKRIVANJE POGREŠAKA

U digitalnim sustavima signali se u pravilu prenose kanalom spojenim na dulje ili kraće veze. Unatoč velikoj pouzdanosti digitalnih sklopova, pri prijenosu informacija povremeno se pojavljuju pogreške i moraju se otkriti.

Za siguran prijenos kôdiranih podataka koristi često se *metoda pariteta*. Svakoj binarnoj kombinaciji kojom se prikazuje digitalni podatak dodaje (barem) jedan bit. Dodatan bīt naziva se *paritetan bīt* (*parity bit*), a svojstvo mu je da u kombinaciji s povećanim brojem bitova uvijek postoji paran broj jedinica *paran paritet* (*even parity*) ili neparan broj jedinica - *neparan paritet* (*odd parity*). Pogreška u

prijenosu što uzrokuje promjenu jednoga bita, mijenja paritet jedinica u podatku, a to upućuje na pogrešku.

33. Primjer: Kôd s parnim paritetom.

Slovu *A* kôdiranome ASCII kôdom pridružena je binarna kombinacija 1000001. Ova kombinacija ima *paran* broj jedinica. Stoga je dodatan paritetan bĩt jednak 0 pa je *A* u ASCII kôdu s *parnim* paritetom 01000001, što opet ukupno daje *paran* broj jedinica.

Slovo *C* kôdirano ASCII kôdom je 1000011, što daje *neparan* broj jedinica. Stoga je paritetan bit 1 pa je *C* u ASCII kôdu s parnim paritetom 11000011, što ukupno daje *paran* broj jedinica.

34. Primjer: Kôd s neparnim paritetom.

Slovo *A* u ASCII kôdu je 1000001, dakle sadrži *paran* broj jedinica. Prema tome, u kôdu s *neparnim* paritetom *A* će biti 11000001, što ukupno daje *neparan* broj jedinica.

Slovo *C* u ASCII kôdu je 1000011, dakle sadrži *neparan* broj jedinica. U kôdu s *neparnim* paritetom *C* će biti 01000011, što opet daje *neparan* broj jedinica.

1.2.7. 1.2.7. KODOVI ZA ISPRAVAK POGREŠAKA

Kodovi za ispravak pogrešaka omogućuju točno odredi mjesto pogreške u podatku. Takvi kôdovi sastoje se od određenoga broja informacijskih bitova z i ispitnih bitova i. Mjesto pogreške otkriva se višestrukim ispitivanjem pariteta određenih kombinacija informacijskih i ispitnih bitova. Takvim ispitivanjem dobije se *ispitan broj* koji pokazuje mjesto pogreške.

Ispitni broj od i bitova može pokazati pogrešku na ukupno $(2^n–1)$ položaja bitova. Što je veći broj znakovnih bitova, potreban je veći broj ispitnih bitova. Kako se ukupna kôdna kombinacija sastoji od z znakovnih i i ispitnih bitova, da bi se u takvome kôdu moglo utvrditi mjesto pogreške mora biti ispunjen uvjet:

$$2^i \geq z + i + 1$$

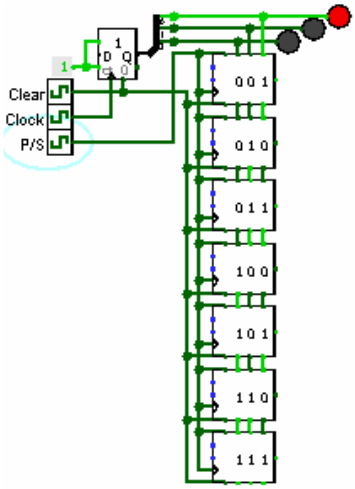
Kao primjer takvoga kôda razmotrit će se Hammingov kôd za dekadske znamenke (tablica 1.16.).

Tablica 1.16. Hammingov kôd za dekadske znamenke

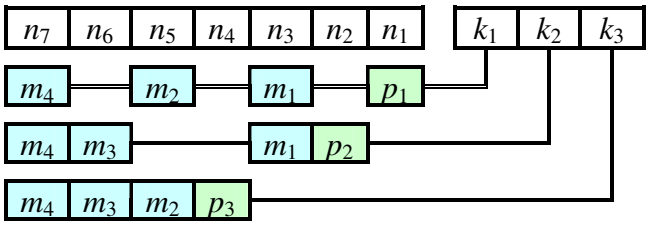
Položaj bita	n_7	n_6	n_5	n_4	n_3	n_2	n_1
Bitovi kôda	m_4	m_3	m_2	p_3	m_1	p_2	p_1
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0

$$\mathbf{H'} = \mathbf{H}^T =$$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 2^2 & 2^1 & 2^0 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}\right] \end{array} \begin{array}{l} \\ \rightarrow 1 \rightarrow 2^0 = 1 \cong 1 \\ \rightarrow 1 \rightarrow 2^1 = 2 \cong 2 \\ 2^0 + 2^1 \cong 3 \\ \rightarrow 1 \rightarrow 2^2 = 4 \cong 4 \\ 2^2 + 2^0 \cong 5 \\ 2^2 + 2^1 \cong 6 \\ 2^2 + 2^1 + 2^0 \cong 7 \end{array}$$



Znakovni bitovi m_1, m_2, m_3 i m_4 služe za prikaz dekadskih znamenaka, a ispitni bitovi p_1, p_2 i p_3 su dodatni bitovi za paritet. Ispitivanje pariteta izvodi se tri puta.

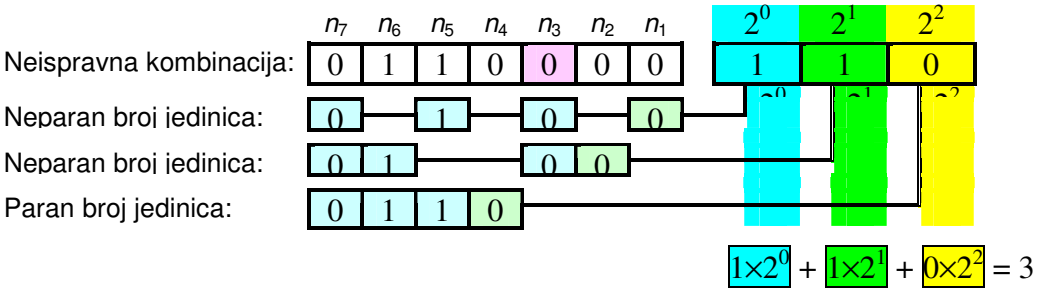


U prvome ispitivanju ispituju se bitovi n_7, n_5, n_3 i n_1, a rezultat se bilježi u k_1. U drugome ispitivanju ispituju se bitovi n_7, n_6, n_3 i n_2, rezultat se bilježi u k_2. U trećemu ispitivanju ispituju se bitovi n_7, n_6, n_5 i n_4, a rezultat se bilježi u k_3. Ako je ispitivanje pariteta uspješno, tj. u ispitivanoj kombinaciji ima paran broj jedinica, onda se rezultat ispitivanja označava s 0. Kada ispitivanje pariteta nije uspješno, tj. kada je u ispitivanoj kombinaciji neparan broj jedinica, onda se rezultat ispitivanja označava s 1.

Na taj način dobije se kombinacija od 3 bita koja označava mjesto bita n na kojemu se nalazi pogreška. Ako su sva tri ispitivanja uspješna, svi k bitovi su 0, što znači da nema pogreške u kôdnim bitovima.

35. Primjer: Treba ispraviti pogrešku u kôdnoj kombinaciju 0110000 sustava

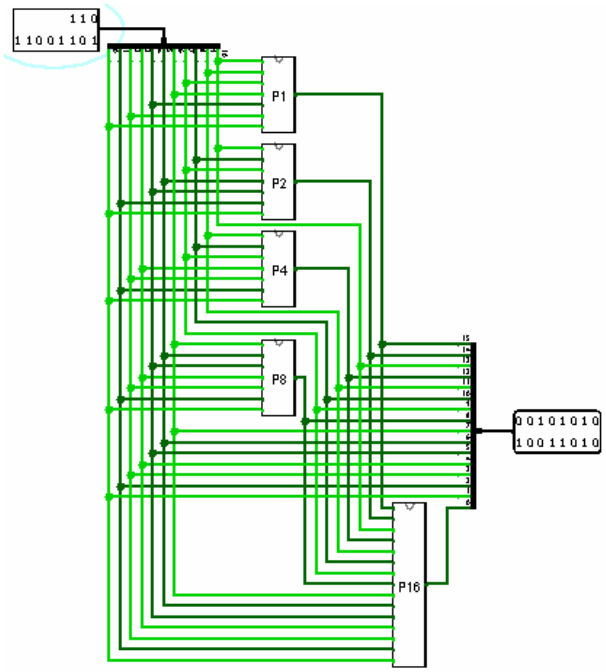
Sustav radi u Hammingovome kôdu.



Pogreška je na mjestu bita n_3, što znači da je ispravna kôdna kombinacija 0110100.

36. Primjer: Izračun pariteta za Hammingov (15, 11) kôd

Ovaj krug ima 11 ulaznih bitova i isprepletene paritetne bitove po potrebi za stvoriti broj od 16 bitova s Hammingovim paritetom. Također, ukupan paritetan bit (bit 16) računa se i pridružuje Hammingovome kodu.



1.2.8. 1.2.8. PREGLED KLJUČNIH POJMOVA

alfanumerički kôdovi (*Alphanumeric Codes*)

- kodovi kojima je prikazuju znakovi, znamenke i slova

Aikenov kôd

- težinski kôd i kôd koji sam stvara komplement, s težinama mjesta 2421.

ASCII kôd (*American Standard Code for Information Interchange*)

- alfanumerički kôd od 7 bitova

BCD (*Binary Coded Decimal*) kôd

- težinski kôd od 4 bita s težinama mjesta 8421

EBCDI kôd (*Extended BCD Interchange Code*)

- alfanumerički kôd od 8 bitova

Excess-3 (XS-3) kôd

- kôd od 4 bita koji sam stvara komplement

Grayev kôd

- kôd od 4 bita u kojemu se iduća kombinacija razlikuje od prethodne samo za jedan bit, tzv. zrcalni kôd

Hammingov kôd

- kôd koji omogućuje ispravak jednostruke pogreške

neparan paritet (*odd parity*)

- dodavanje paritetnoga bita binarnoj kombinaciji kôda tako da je ukupan broj jedinica u kombinaciji neparan

paran paritet (*even parity*)

- dodavanje paritetnoga bita binarnoj kombinaciji kôda tako da je ukupan broj jedinica u kombinaciji paran

paritetan bit (*parity bit*)

- dodatan bit koji se dodaje osnovnoj kombinaciji nekoga kôda radi otkrivanja moguće pogreške u prijenosu

zrcalni ili odrazni binarni brojevni sustavi (*reflected binary number systems*)

- binaran brojevni sustavi u kojima se brojevi redom razlikuju od prethodnoga za jedan bit, oni su osnova Grayeva kôda

kôd koji sam stvara komplement (*self-complemented code*)

- kôd u kojemu se kombinacija za komplement bilo koje znamenke dobije jednostavnom zamjenom nula jedinicama i obrnuto

težinski kôd (*weighted code*)

- kôd u kojemu znamenke kombinacije imaju određene težine mjesta

Tablica 1.17. Pregled kôdova sazdanih od 4 bita

	Dekadska kombinacija u kôdu			
Binarna kombinacija	BCD	XS-3	Aikenov	Grayev
0000	0		0	0
0001	1		1	1
0010	2		2	3
0011	3	0	3	2
0100	4	1	4	7
0101	5	2		6
0110	6	3		4
0111	7	4		5
1000	8	5		
1001	9	6		
1010		7		
1011		8	5	
1100		9	6	8
1101			7	9
1110			8	
1111			9	

Tablica 1.18. Stari ASCII kôd sa slovima hrvatske abecede (pojedini rjeđe korišteni znakovi koristili su se zamjenski za 10 hrvatskih specifičnih malih i velikih slova)

				b7	0	0	0	0	1		1		1		1	
				b6	0	0	1	1	0		0		1		1	
				b5	0	1	0	1	0		1		0		1	
b4	b3	b2	b1		0	1	2	3	4		5		6		7	
0	0	0	0	0	NUL	DLE	SP	0	Ž	@	P		ž	`	p	
0	0	0	1	1	SOH	DC1	!	1	A		Q		a		q	
0	0	1	0	2	STX	DC2	"	2	B		R		b		r	
0	0	1	1	3	ETX	DC3	#	3	C		S		c		s	
0	1	0	0	4	EOT	DC4	$	4	D		T		d		t	
0	1	0	1	5	ENQ	NAK	%	5	E		U		e		u	
0	1	1	0	6	ACK	SYN	&	6	F		V		f		v	
0	1	1	1	7	BEL	ETB	'	7	G		W		g		w	
1	0	0	0	8	BS	CAN	(	8	H		X		h		x	
1	0	0	1	9	HT	EM	)	9	I		Y		i		y	
1	0	1	0	A	LF	SUB	*		J		Z		j		z	
1	0	1	1	B	VT	ESC	+	;	K		Š	[	k		š	{
1	1	0	0	C	FF	FS	,	<	L		Đ	\	l		đ	|
1	1	0	1	D	CR	GS	-	=	M		Ć	]	m		ć	}
1	1	1	0	E	SO	RS	>		N		Č	^	n		č	~
1	1	1	1	F	SI	US	/	?	O		_		o		DEL	

1.2.9. 1.2.9. PITANJA I ZADACI ZA PONAVLJANJE

1. U čemu se razlikuju BCD kôd i Excess-3 kôd?

2. Objasnite pojmove težinski i samo-komplementirajući kôd.

3. Kôdirajte BCD kôdom broj 395.

   0011 1001 0101

4. Koji broj dekadskoga brojevnoga sustava odgovara binarnoj kombinaciji 100001100010 u BCD kôdu?

   862

5. Koliko binarnih znamenaka treba da se broj 128_{10} napiše u binarnome, a koliko u BCD kôdu?

   8, 12

6. Kôdirajte XS-3 kôdom broj 395.

   0110 1100 1000

7. Koji broj dekadskoga brojevnoga sustava odgovara binarnoj kombinaciji 101110010101 u XS-3 kôdu?

   862

8. Kôdirajte Grayevim kôdom broj 286.

   0011 1100 0101

9. Koji broj dekadskoga brojevnoga sustava odgovara binarnoj kombinaciji 001011010111 u Grayevu kôdu?

   395

10. Kôdirajte Aikenovim kôdom broj 728.

    1101 0010 1110

11. Koji broj dekadskoga brojevnoga sustava odgovara binarnoj kombinaciji 110001001111 u Aikenovu kôdu?

    649

12. Kôdirajte podatak Y : 5 u ASCII kôdu.

    1011001 0111010 0111001

13. Koji je sadržaj podatka 1011000 0101011 0111000 zadanog u ASCII kôdu?

    X + 8

14. Kôdirajte podatak Y : 8 u EBCDI kôdu.

    11101000 01111010 11111000

15. Koji je sadržaj podatka 11100010 11010111 01001110 zadanoga u EBCDI kôdu?

    SP+

16. Objasnite funkciju paritetnoga bita.

17. U tablici ASCII kôda pronađite binarnu kombinaciju za slova *s* i *S* i odredite vrijednost paritetnoga bita prema uvjetima za paran paritet?

    1 1110011, 0 1010011

18. U tablici ASCII kôda pronađite binarnu kombinaciju za slova *p* i *P* i odredite vrijednosti paritetnoga bita prema uvjetima za neparan paritet?

    0 1110000, 1 1010000

19. Nađite ispravnu kombinaciju za podatak 0011001 0111101 zadan Hammingovim kôdom.

    0011001 0101101

2. 2. LOGIČKI SKLOPOVI

Digitalni sklopovi mogu imati jedan ili više ulaza i isto toliko izlaza. Naponi na ulazima i izlazima mogu imati vrijednosti unutar područja koja odgovaraju binarnim znamenkama 0 ili 1. Stanje napona na izlazima sklopova vezano je za ispunjenje određenih uvjeta na ulazima. Između stanja na ulazima i stanja izlaza postoji određena logička veza, odnosno digitalni sklopovi obavljaju logičke funkcije ili operacije. Stoga se digitalni sklopovi nazivaju i logički sklopovi.

Kombinacijski logički sklopovi nazivaju se logički sklopovi čije stanje izlaza ovisi o trenutnome stanju ulaza. *Serijski* (slijedni) sklopovi nazivaju se sklopovi čije stanje izlaza ovisi o stanju ulaza i o prethodnome stanju na izlazu.

Digitalni sklopovi u shemama digitalnih uređaja, prikazuju se odgovarajućim simbolima. Vrlo često koriste se simboli prema američkim standardima (MIL-ST-806B 1962. *Graphic Symbols for Logic Diagrams, Department of Defense, USA*). Od 1984. godine koriste se simboli prema standardima IEC (*International Electro-technical Commission*).

Tablica stanja (*truth table*) iskazuje logička svojstva digitalnih sklopova. Ona je pregledan prikaz svih kombinacija ulaznih binarnih veličina i odgovarajućih stanja na izlazu. U tvorničkim podacima proizvođača digitalnih sklopova često umjesto oznaka 0 i 1 koriste oznake L (*low* = nisko) i H (*high* = visoko).

Engleski matematičar George Boole razvio je u 19. stoljeću *logičku algebru* (danas poznatu pod imenom *Booleova algebra*) što se koristi za analizu i sintezu logičkih sklopova. Tako se logička svojstva digitalnih sklopova mogu iskazati i algebarskim ili logičkim jednadžbama.

U ovome poglavlju razmatra se međusobno povezivanje osnovnih logičkih sklopova, koristeći složenije logičke operacije logičke algebre.

2.1. 2.1. OSNOVNI LOGIČKI SKLOPOVI

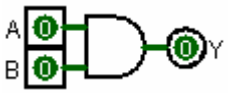
U ovome poglavlju razmatraju se logička svojstva osnovnih logičkih sklopova i njihovo spajanje zbog izvođenja složenijih logičkih operacija. Električna svojstva tih sklopova iscrpno se obrađuju u poglavlju o skupinama integriranih logičkih sklopova. Postoje 3 osnovna logička sklopa I, ILI i NE. Njihove suprotnosti su NI, NILI i NE, a jedna i druga skupina, povezana u različite kombinacije zadovoljavaju sve logičke operacije Booleove algebre.

2.1.1. 2.1.1. LOGIČKI I SKLOP

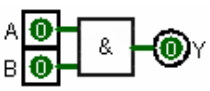
Logičku operaciju I (povezivanje, konjunkcija) ostvaruje logički sklop I (*AND gate*). Logički sklop I predstavlja matematičku operaciju množenja. Sklop ima dva ili više ulaza, a na izlazu daje stanje 1 samo onda ako su svi ulazi u stanju 1. Ako je na bilo kojemu ulazu sklopa logičko stanje 0, onda je i izlaz u stanju 0.

Logički sklop I s 2 ulaza

(a) ANSI simbol



(b) IEC simbol



(c) tablica stanja

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d) logičke operacije

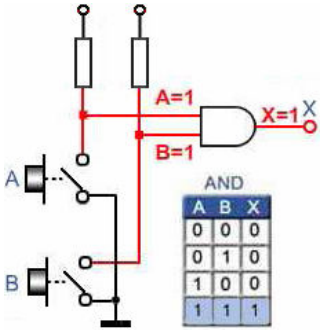
$0 \cdot 0 = 0$
$0 \cdot 1 = 1$
$1 \cdot 0 = 1$
$1 \cdot 1 = 1$

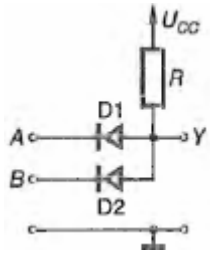
(f) algebarski izraz

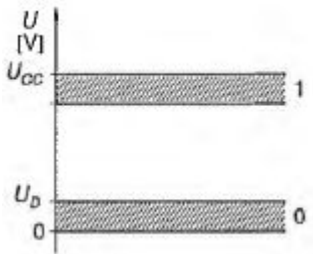
$Y=A \cdot B=A \wedge B=A\&B$

Sklop ovih svojstava moguće je izvesti spajanjem otpornika i dioda kao na slici.

Izvedba sklopa I s dva ulaza



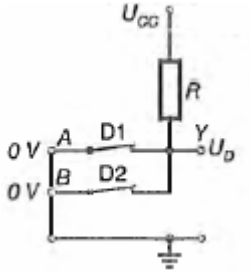




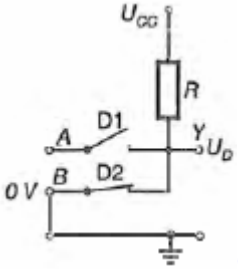
A [V]	B [V]	Y [V]
0	0	U_D
U_{CC}	0	U_D
0	U_{CC}	U_D
U_{CC}	U_{CC}	U_{CC}

Da bi izlaz Y bio na logičkoj razini 1, niti jedna dioda ne smije se spojiti na uzemljenje. Ako je na oba ulaza napon 0 V (uzemljene diode), to odgovara logičkome stanju 0, obje diode su propusno polarizirane. Stoga je na izlazu Y, malen napon U_D (pad napona na propusno polariziranoj diodi), što također odgovara logičkome stanju 0. Obje diode u ovome slučaju predstavljaju uključene sklopke pa se sklop jednostavno može prikazati shemom.

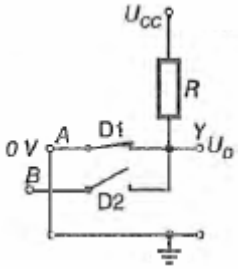
Pojednostavnjen logički I sklop i nadomjesna shema dioda(ma)



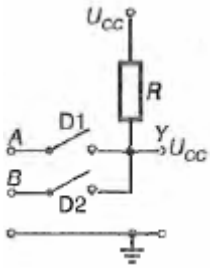
(a)



(b)



(c)

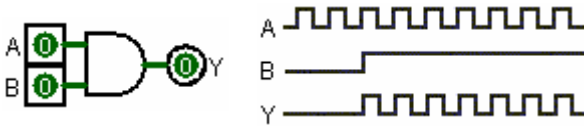


(d)

U prikazu je zanemaren pad napona na diodi U_D. Ako se ulaz (A i/ili B) spoji na 0 V, dioda je propusno polarizirana. Napon na izlazu Y ostaje U_D sve dok je barem jedan od ulaza spojen na potencijal 0 V, a to simbolizira uključenu sklopku. Jedino ako su obje sklopke otvorene, izlaz Y je u logičkome stanju 1, jer se napon U_{CC} preslikava na izlaz.

Nepostojanje ulaznoga napona ili dovođenje napona U_{CC} na ulaz, zaporno polarizira diode pa one predstavljaju isključene sklopke. Izlazni napon ima praktično vrijednost U_{CC}, što odgovara logičkome stanju 1. Na prikazanim ulaznim točkama, diode simboliziraju sklopke, jer ako je na ulaznim točkama napon 0, diode vode pa u bilo kojoj kombinaciji "prekidača", napon na izlazu Y simbolizira logičku 0. Logički sklop I može se koristiti kao sklop za dozvolu (*enable*) i zabranu (*inhibit*) prolaza impulsa.

Dozvola i zabrana prolaza impulsa realizirana logičkim I sklopom

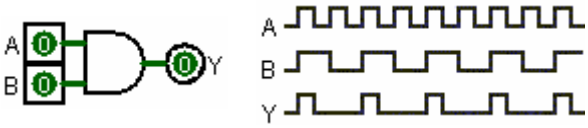


Signalu s ulaza A dozvoljen je prolaz do izlaza samo ako je drugi ulaz sklopa I u stanju 1.

37. Primjer: Odrediti oblik impulsa na izlazu sklopa I uz zadane signale na ulazima A i B

Izlaz I sklopa Y u stanju je 1 onda i samo onda, ako su oba ulaza *istovremeno* u stanju 1.

Odziv sklopa I na impulsnu pobudu

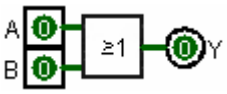


2.1.2. 2.1.2. LOGIČKI ILI SKLOP

Logičkoj operaciji ILI (rastavljanje, disjunkcija) svojstvena je matematička operacija zbrajanja, a ostvaruje ju logički sklop ILI (OR gate). Sklop imati dva ili više ulaza, a na izlazu je stanje 1 samo onda ako je na bilo kojemu ulazu stanje 1 ili je na oba ulaza stanje 1. Ako su svi ulazi u stanju 0, izlaz je u stanju 0.

Logički sklop ILI s 2 ulaza

(a) ANSI simbol (b) IEC simbol (c) tablica stanja (d) logičke operacije (e) algebarski izraz



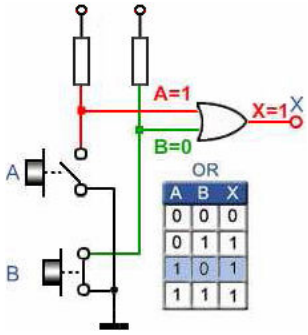
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

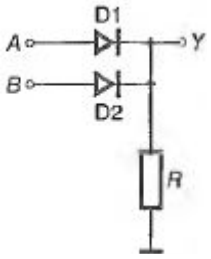
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

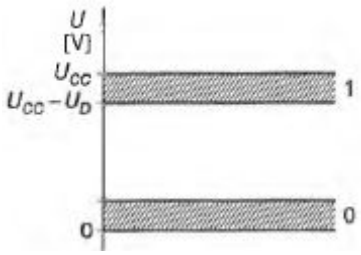
$X = A + B = A \vee B$

Sklop na slici (spoj jednoga otpornika i dvaju dioda) ima ista svojstva kao i prethodan sklop.

Logički ILI sklop s 2 ulaza (izvedba diodama)





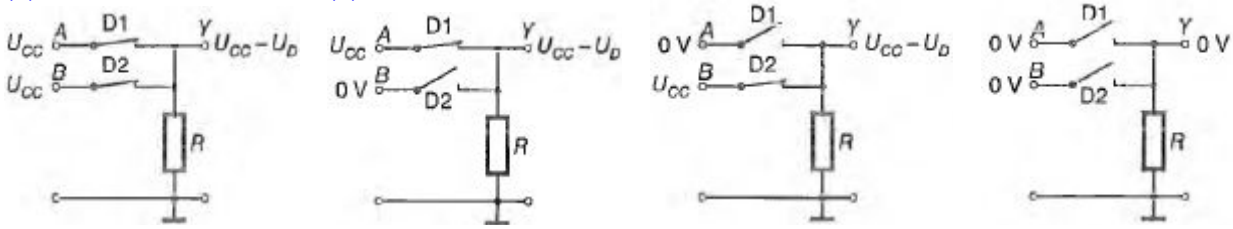


A [V]	*B* [V]	Y [V]
0	0	0
U_{CC}	0	U_{CC}–U_D
0	U_{CC}	U_{CC}–U_D
U_{CC}	U_{CC}	U_{CC}–U_D

Ovisno o naponima na ulazima A i B, diode su propusno ili nepropusno polarizirane.

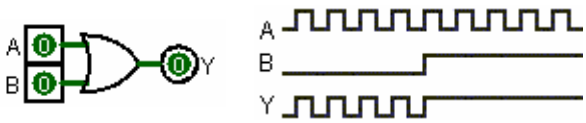
Pojednostavnjena nadomjesna shema logičkoga ILI sklopa

(a) (b) (c) (d)



Ako je napon U_{CC} na ulazima A i B, to odgovara logičkoj 1, jer se obje diode propusno polariziraju (uključene sklopke) pa se na izlaz Y preslikava napon U_{CC} umanjen za pad napona na propusno polariziranim diodama (PAZI: paralelan spoj!). U slučajevima jedne nepropusno polarizirane diode, na isti način preko one druge diode, na izlaz Y preslikava se napon U_{CC}. Ako na oba ulaza nema propusno polarizirajućega napona U_{CC}, onda se preko otpornika R, napon od 0 V preslikava na izlaz Y i on je u logičkome stanju 0. Logički sklop ILI može se također koristiti kao sklop za dopuštenje i zabranu prolaza impulsa.

Dozvola i zabrana prolaza impulsu ostvarena logičkim ILI sklopom

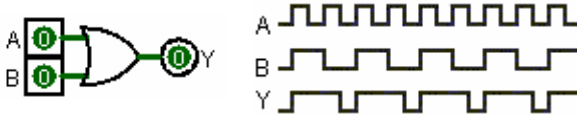


Signal s ulaza A nalazi se na izlazu samo kada je drugi ulaz u stanju 0.

38. Primjer: Odrediti oblik impulsa na izlazu sklopa ILI uz zadane signale na ulazima A i B

Ako su svi ulazi u stanju 0, onda je izlaz Y sklopa ILI u stanju 0. To znači da se na izlazu Y dobije impuls u trenucima ako na bilo kojemu od ulaza ILI sklopa ima impuls.

Odziv logičkoga ILI sklopa na impulsnu pobudu oba ulaza

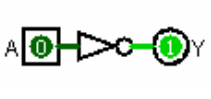


2.1.3. 2.1.3. LOGIČKI NE SKLOP

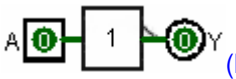
Logički sklop NE, odnosno *invertor* (*NOT circuit, inverter*) obavlja logičku operaciju NE (negacija, inverzija, komplement). Sklop ima jedan ulaz i jedan izlaz. Na izlazu daje stanje suprotno stanju ulaza. Ako je na ulazu stanje 1, na izlazu je stanje 0 i obrnuto.

Logički sklop NE

ANSI simbol | IEC simbol | tablica stanja | ulazni valni oblici | logičke operacije | algebarski izraz



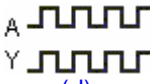
(a)



(b)

A	Y
0	1
1	0

(c)



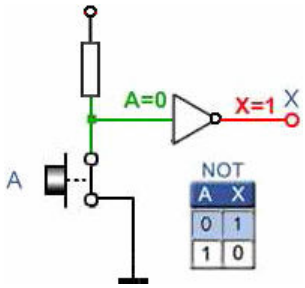
(d)

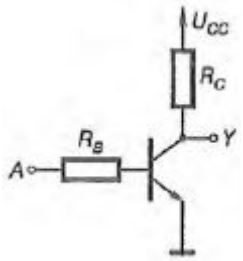
$\overline{0} = 1,\ \overline{1} = 0$ (f)

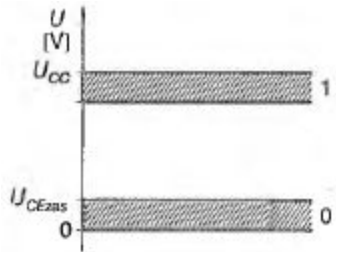
$Y = \overline{A}$ (e)

Funkciju logičkoga NE sklopa, prikazuje tranzistorska sklopka.

Izvedba logičkoga NE sklopa



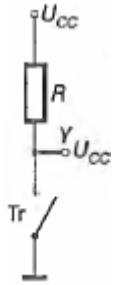




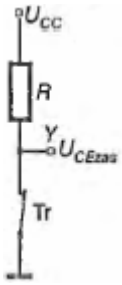
A [V]	Y [V]
0	U_{CC}
U_{CC}	U_{CCzas}

Ako je na ulazu sklopa u točki A napon 0 V (logičko stanje 0), radna točka tranzistora je u zapornome području i on ne vodi značajnu struju pa predstavlja isključenu sklopku. Na izlazu Y napon je jednak U_{CC}, a to je logičko stanje 1.

Pojednostavnjena nadomjesna shema logičkoga NE sklopa



(a)



(b)

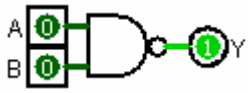
Ako je na ulazu tranzistorske sklopke u točki A, napon U_{CC}, znači da tranzistor vodi i radna točka tranzistora je u zasićenju. Tranzistor predstavlja uključenu sklopku. Na izlazu je malen napon U_{CEzas}, a to na izlazu Y predstavlja logičko stanje 0.

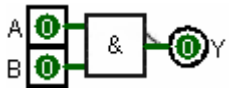
2.1.4. 2.1.4. LOGIČKI NI SKLOP

Logički sklop NI (NOT AND, NAND gate) obavlja logičku operaciju negacije. Ovaj sklop poznat je i pod nazivom Shefferova[4] funkcija). Sklop ima dva i više ulaza. Ako je na bilo kojemu ulazu stanje 0, izlaz je u stanju 1. Ako je na svim ulazima stanje 1, tada je na izlazu stanje 0.

Logički NI sklop s 2 ulaza

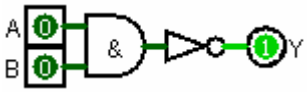
(a) ANSI simbol | (b) IEC simbol | (c) algebarski izraz | (d) tablica stanja | (e) ekvivalentan sklop





$Y = \overline{A \cdot B}$

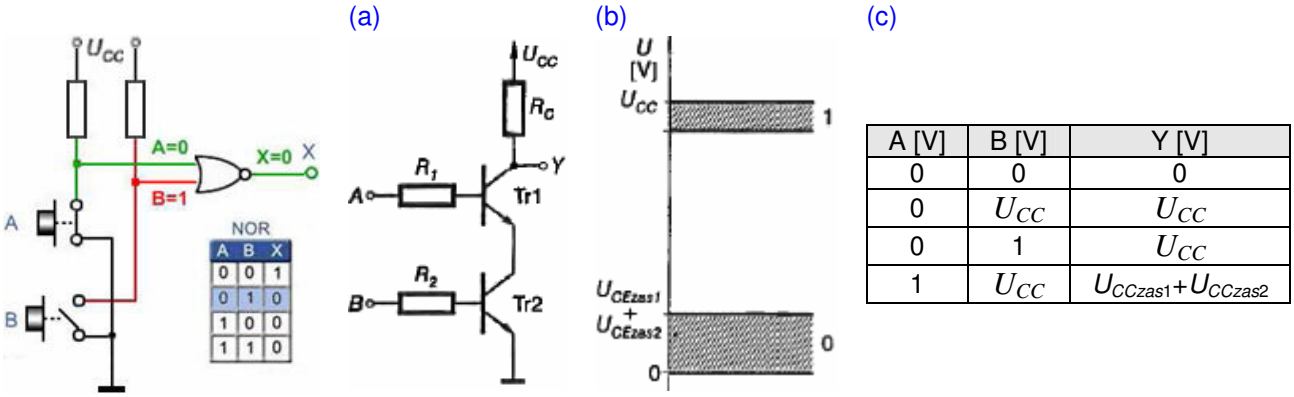
A	B	Y
0	0	1
1	0	1
0	1	1
1	1	0



Na temelju izloženih logičkih svojstava sklopa NI jasno je da se sklop može izvesti spajanjem diodnoga sklopa I i tranzistorske sklopke. No sklop je moguće izvesti i serijskim spojem dviju tranzistorskih sklopki.

[4] Simbol poteza-crte (*stroke*) "↑" nazvan je prema Henry M. Sheffer, koji je 1913. objavio rad u časopisu *Transactions of the American Mathematical Society* (Sheffer 1913) i omogućio aksiomatizaciju Booleovih izraza (Boolean algebras) koristeći okomitu crtu sa strelicom na vrhu (*stroke*) te dokazao njezinu ekvivalenciju standardnoj formulaciji koju je napravio Huntington primjenjujući već znane operatore poznate logike (AND, OR, NOT)

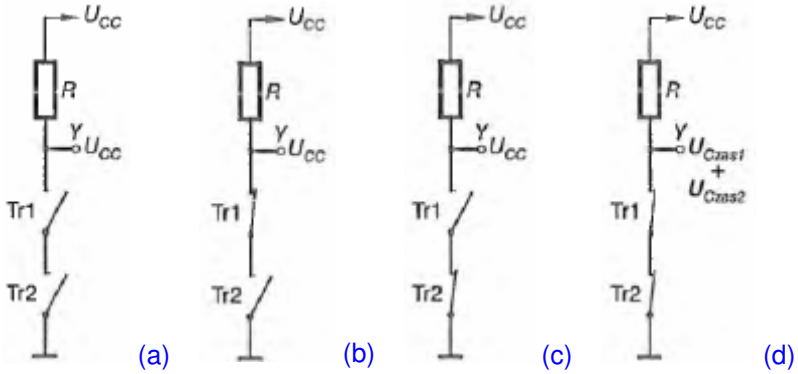
Izvedba logičkoga NI sklopa



A [V]	B [V]	Y [V]
0	0	0
0	U_{CC}	U_{CC}
0	1	U_{CC}
1	U_{CC}	$U_{CCzas1}+U_{CCzas2}$

Ako je na oba ulaza sklopa A i B, napon 0 V, radne točke oba tranzistora u zapornome su području. One djeluju kao isključene sklopke pa je na izlazu Y napon U_{CC}, dakle logičko stanje 1. Ovakvo stanje ostaje na izlazu sve dok je na bilo kojemu od ulaza napon 0 V, jer je pripadni tranzistor isključena sklopka.

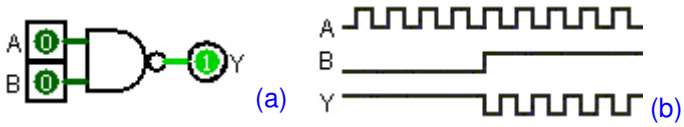
Pojednostavnjena nadomjesna shema logičkoga NI sklopa



Iz nacrtanih simboličkih shema na slici, ako oba tranzistora vode, znači da je na oba ulaza napon U_{CC} (logičko stanje 1). Oba tranzistora prelaze u stanje zasićenja, dakle djeluju kao uključene sklopke pa je na izlazu malen napon zasićenja obaju tranzistora $U_{CEzas1}+U_{CEzas2}$, tj. logičko stanje 0.

Ako se na jedan ulaz sklopa NI dovede niz impulsa, signal s toga ulaza pojavit će se u invertiranome obliku na izlazu samo ako je drugi ulaz u stanju 1. Ako je drugi ulaz u stanju 0, tada je i izlaz Y u stanju 1 bez obzira na signal s prvoga ulaza.

Zabrana i dozvola prolaza impulsu ostvarena logičkim NI sklopom

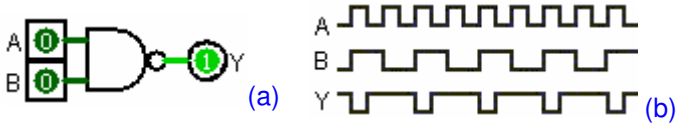


To znači da i sklop NI može poslužiti kao sklop za zabranu i dopuštenje prolaza impulsa. Pri tome treba imati na umu da se pri dopuštenju prolaza impulsi invertiraju.

39. Primjer: Odredite oblik napona na izlazu sklopa NI uz zadane signale na ulazima A i B

Ako su svi ulazi u stanju 1, onda je izlazni napon sklopa NI u stanju 0. To znači da se na izlazu Y dobije impuls u periodu ako ne postoji impuls na bilo kojemu od ulaza.

Odziv logičkoga NI sklopa na impulsne pobude ulaza

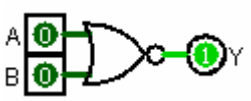


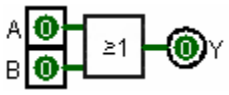
2.1.5. 2.1.5. LOGIČKI NILI SKLOP

Logički sklop NILI (*NOR gate, NOT OR*) obavlja logičku operaciju NILI (Piercova funkcija[5]). Sklop može imati dva ili više ulaza. Na izlazu je stanje 1 samo ako su svi ulazi u stanju 0. Ako je na bilo kojemu ulazu stanje 1, na izlazu je stanje 0.

Logički NILI sklop s 2 ulaza

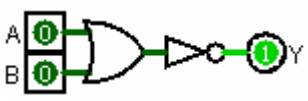
(a) ANSI simbol (b) IEC simbol (c) algebarski izraz (d) tablica stanja (e) ekvivalentni sklop





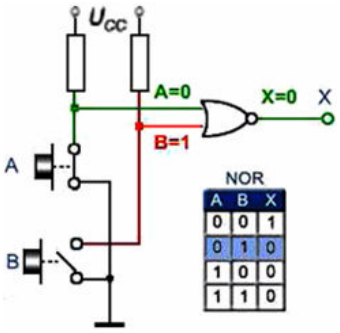
$Y = \overline{A + B}$

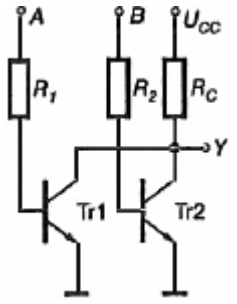
A	B	Y
0	0	1
1	0	0
0	1	0
1	1	0

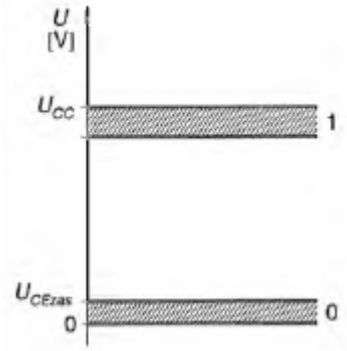


Na temelju izloženih logičkih svojstava sklopa NILI jasno je da se sklop može izvesti spajanjem diodnoga sklopa ILI i tranzistorske sklopke. Sklop je moguće izvesti i paralelnim spojem dviju tranzistorskih sklopki.

Izvedba logičkoga NILI sklopa tranzistorima







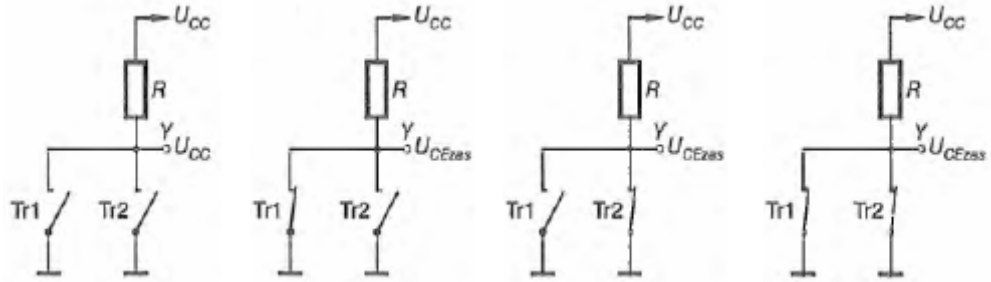
A [V]	B [V]	Y [V]
0	0	U_{CC}
U_{CC}	0	U_{CCzas}
0	U_{CC}	U_{CCzas}
U_{CC}	U_{CC}	U_{CCzas}

(a) (b) (c)

Ako je na oba ulaza sklopa napon 0 V, radne točke oba tranzistora su u području zapiranja, dakle djeluju kao isključene sklopke pa je na izlazu Y napon U_{CC}, što odgovara logičkome stanju 1.

Pojednostavnjena nadomjesna shema logičkoga NILI sklopa

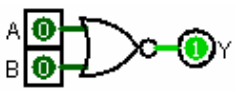
(a) (b) (c) (d)



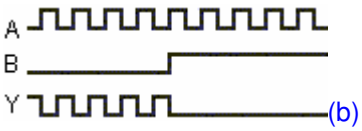
Ako se na bilo koji od ulaza sklopa dovede napon U_{CC} (logičko stanje 1), pripadni tranzistor provede te prelazi u stanje zasićenja. Takav tranzistor djeluje kao uključena sklopka, pa je izlaz spojen na uzemljenje (odnosno mali napon zasićenja tranzistora U_{CEzas}). Dakle to je za izlaz Y logičko stanje 0.

Ako se na ulaz A sklopa NILI dovede niz impulsa, signal s toga ulaza pojavit će se u invertiranome obliku na izlazu samo ako je drugi ulaz B u stanju 0. Ako je drugi ulaz B u stanju 1, izlaz Y je u stanju 0 bez obzira na signal s prvoga ulaza A.

Dozvola i zabrana prolaza impulsu pomoću logičkoga NILI sklopa



(a)



(b)

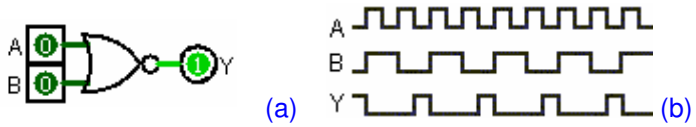
[5] Charles Sanders Peirce (1880) otkrio je funkcijsku cjelovitost za NAND ili NOR više od 30 godina prije navedene godine, koristeći izraz *ampheck* za "rezanje u oba smjera" (*cutting both ways*), ali to svoje otkriće nikada nije objavio.

To znači da se i sklop NILI može koristiti kao sklop za dopuštenje ili zabranu prolaza impulsa. Ako je dopušten prolaza impulsa, signali se invertiraju.

40. Primjer: Odrediti oblik napona na izlazu sklopa NILI uz zadane signale na ulazima A i B

Izlazni napon sklopa NILI je u stanju 1 samo ako su oba ulaza u stanju 0. To znači da se na izlazu Y dobije impuls samo u trenutku kada na oba ulaza *istodobno nema* impulsa.

Odziv logičkoga NILI sklopa na ulaznu impulsnu pobudu

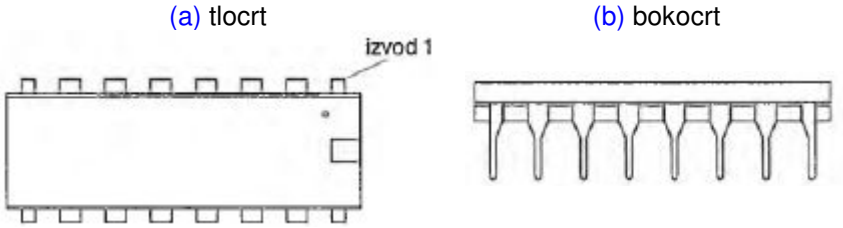


2.1.6. 2.1.6. INTEGRIRANI LOGIČKI SKLOPOVI

Elektronički logički sklopovi proizvode se u integriranoj izvedbi. Sve komponente sklopa izvedene su na jednoj pločici silicija zatvorenoj u odgovarajuće kućište. Kućišta mogu biti plastična ili keramička različitoga oblika s različitim brojem izvoda. Sama kućišta znatno su većih dimenzija od integriranoga sklopa. Ovisno o broju ulaza i izlaza logičkoga sklopa, u jedno kućište moguće je smjestiti jedan nekoliko logičkih sklopova.

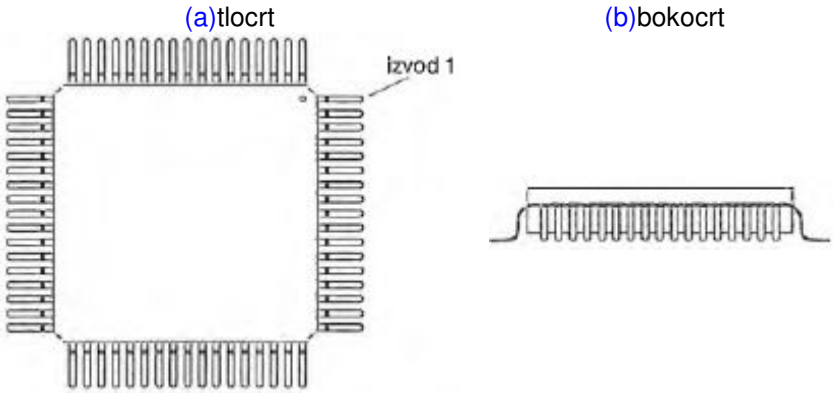
Čest oblik kućišta integriranih sklopova je dvoredno kućište DIP (*dual-in-line package*), zatim plosnato kućište (*flat package*) i sve češće različiti oblici kućišta za površinsku montažu (*surface mounted device, leadless package, leadless chip carrier, small-outline package*).

Dvoredno kućište sa 16 izvoda



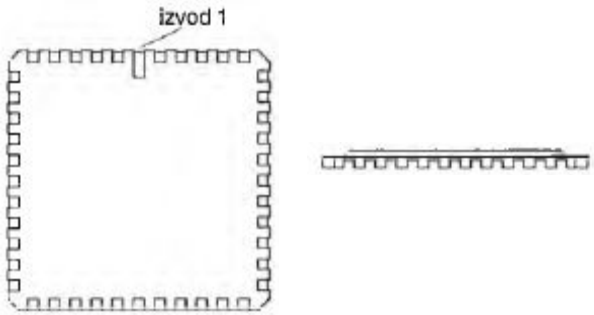
Osnovni integrirani logički sklopovi sadrže manji broj integriranih elemenata (do 100) i nazivaju se sklopovi niskoga stupnja integracije SSI (*Small Scale Integration*). Složeniji integrirani sklopovi (brojila, registri, dekoderi) sadrže veći broj integriranih elemenata (od 100 do 1000) i nazivaju se sklopovi srednjega stupnja integracije MSI (*Medium Scale Integration*). Još veći broj elemenata (od 1000 do 100000) sadrže sklopovi visokoga stupnja integracije LSI (*Large Scale Integration*). U ovu skupinu spadaju memorije i mikroprocesori. Već nekoliko desetljeća pojavljuju se sklopovi vrlo visokoga stupnja integracije VLSI (*Very Large Scale Integration*). To su sklopovi s više od 1000000 integriranih elemenata.

Izvedba plosnatoga kućišta za površinsku montažu



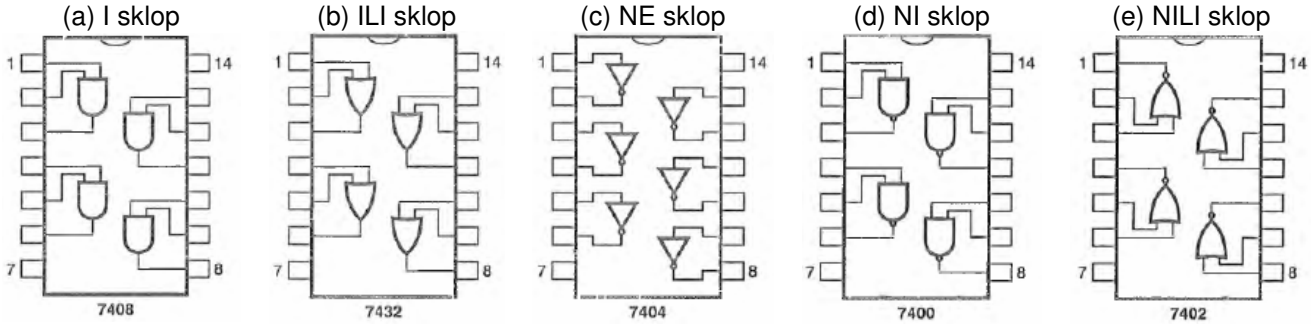
Svi integrirani logički sklopovi mogu se svrstati u nekoliko skupina. Za sklopove unutar neke skupine karakteristično je da su prilagođeni za međusobno spajanje, što omogućuje relativno jednostavnu gradnju složenih digitalnih uređaja.

(a)tlocrt (b)bokocrt



Pri radu s integriranim sklopovima neophodno je poznavati raspored izvoda ili dijagram spajanja (*pin connection diagram, pin assignment, pin description, pin configuration*). Iz njega se vide funkcije izvoda integriranoga sklopa. Postupak brojenja izvoda vidi se iz prikaza tipova pojedinih kućišta. Izvodi označeni s U_{CC}, odnosno V_{CC} (*voltage*) i *GND* (*ground*), služe za spajanje napona napajanja zajedničkoga za sve logičke sklopove unutar jednoga kućišta. Na izvod V_{CC} spaja se pozitivan pol izvora napajanja, a na izvod *GND* negativan pol.

Raspored izvoda integriranih logičkih sklopova, pogled odozgo (7 … GND, 14 … U_{CC})



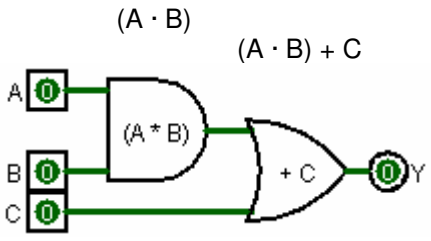
a) I, b) ILI, c) NE, d) NI, e) NILI (7 … GND, 14 … U_{CC})

2.1.7. 2.1.7. MEĐUSOBNO POVEZIVANJE OSNOVNIH LOGIČKIH SKLOPOVA

Osnovni logički sklopovi spajaju se međusobno zbog izvođenja složenijih logičkih operacija. Složena logička operacija može se zadati algebarskim izrazom. Postupnim crtanjem simbola osnovnih logičkih sklopova dobije se odgovarajuća logička shema složenoga logičkoga sklopa.

Složena logička operacija zadana algebarskim izrazom: Y = (A · B) + C

(a) logička shema



(b) tablica stanja

C	B	A	A·B	A (B+C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

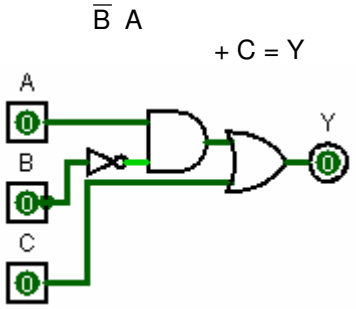
Iz algebarskoga izraza moguće je odrediti tablicu stanja (slika 28.b). U simulacijskome paketu LogiSim, tablica stanja automatski se generira na temelju upisanoga (*copy/paste*) algebarskoga izraza.

41. Primjer: Nacrtati logičku shemu i tablicu istine sklopa koji obavlja operaciju:

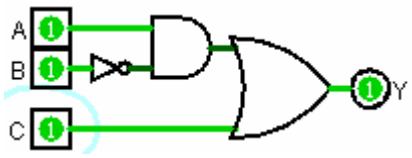
$$Y = A\ \overline{B}\ + C.$$

Tablicom stanja prikazati logička svojstva sklopa. Na temelju tablice stanja, logička shema složenoga logičkoga sklopa stvara se automatski, kao i algebarski izraz. Najprije se upišu izrazi za ulaze i izlaz. Zatim se unese algebarski izraz u obliku što ga prepoznaje LogiSim. Izraz je: A ~B + C i u takvome obliku unosi se u LogiSim. Uočite razmak između faktora i oko znaka +, a za negaciju koristi se simbol ~ (*tilda*).

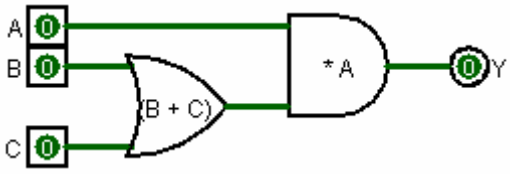
Složena logička operacija zadana algebarskim izrazom: Y = A $\overline{B}$ + C



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



42. Primjer: Za sklop na slici , napišite algebarski izraz i tablicu stanja.



(a) Y = A · (B + C)

A	B	C	B·C	A (B+C)
0	0	0	0	0
1	0	0	0	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	0
1	1	1	1	1

(b)

2.1.8. 2.1.8. PREGLED KLJUČNIH POJMOVA

algebarski izraz za logički sklop, odnosno logička jednadžba sklopa

- algebarski prikaz logičke ovisnosti izlaznoga stanja sklopa o ulaznome stanju

H (*High* … visoko)

- oznaka za visoku naponsku razinu, odnosno stanje 1 na ulazima i izlazima digitalnih sklopova

kombinacijski logički sklop

- logički sklop čije stanje izlaza ovisi o trenutnim stanjima na ulazima

L (*Low* … nisko)

- oznaka za nisku naponsku razinu, odnosno stanje 0 na ulazima i izlazima digitalnih sklopova

logička shema

- shema složenoga logičkoga sklopa u kojoj su jednostavniji logički sklopovi predočeni logičkim simbolima

logički simbol

- grafički simbol za logički sklop

logički sklop

- digitalni sklop koji obavlja logičke operacije, na ulazima i izlazima mogu biti samo naponska stanja koja se prikazuju binarnim znamenkama 0 i 1

logički sklop I (AND gate)

- sklop koji na izlazu daje stanje 1 samo ako su svi ulazi u stanju 1

logički sklop ILI (OR gate)

- sklop koji na izlazu daje stanje 1 ako je bilo koji ulaz u stanju 1

logički sklop NE (NOT *circuit*)

- sklop koji daje na izlazu stanje suprotno stanju na ulazu

logički sklop NI (NAND gate)

- sklop koji daje na izlazu stanje 1 ako je bilo koji ulaz u stanju 0

logički sklop NILI (NOR gate)

- sklop koji daje na izlazu stanje 1 samo ako su svi ulazi u stanju 0

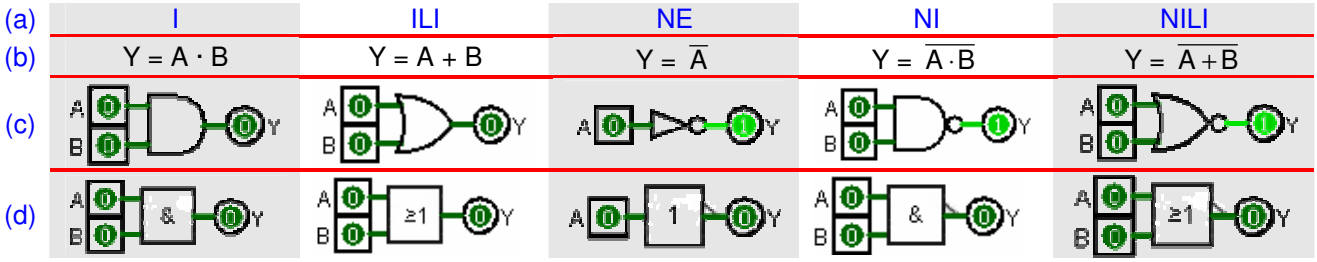
serijski logički sklop

- logički sklop čije stanje izlaza ovisi o trenutnim stanjima na ulazima i prethodnome stanju na izlazu.

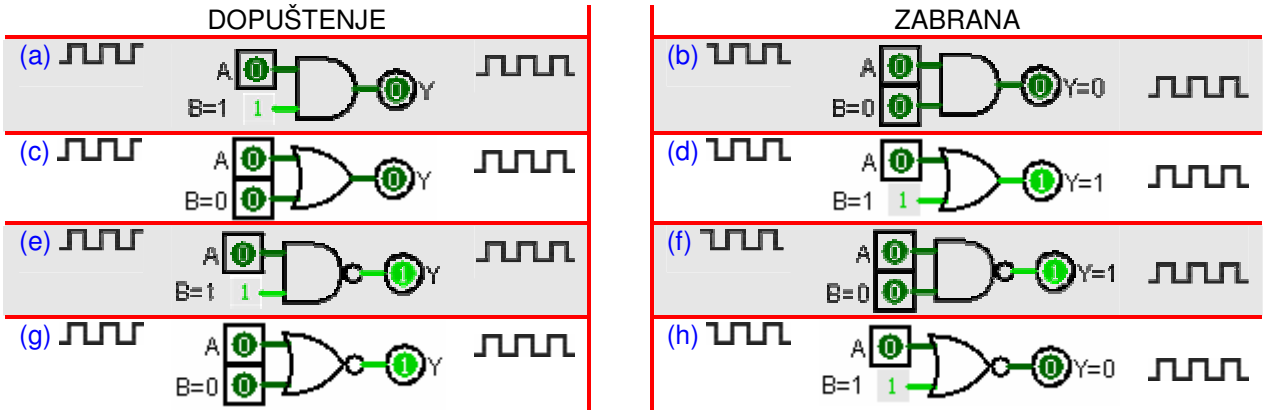
tablica stanja (*truth table*)

- tablični prikaz svih kombinacija ulaznih veličina logičkoga sklopa i odgovarajućih stanja na izlazima

Osnovni logički sklopovi za dozvolu (*enable*) i zabranu (*inhibit*) prolaza ulaznome impulsu:



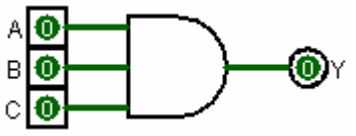
Dopuštenje i zabrana prolaza impulsa pomoću osnovnih logičkih sklopova



2.1.9. 2.1.9. PITANJA I ZADACI ZA PONAVLJANJE

1. Navedite logička svojstva sklopa I.

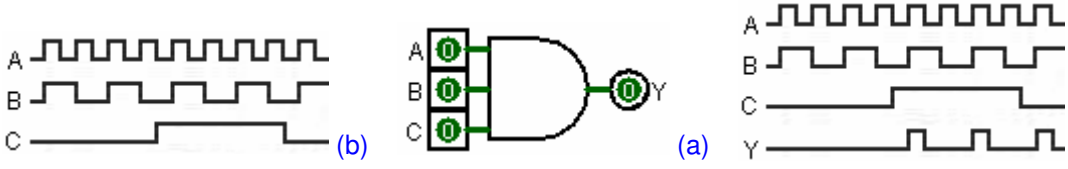
2. Nacrtajte simbol sklopa I s tri ulaza i napišite pripadni algebarski izraz i tablicu stanja.



$Y = A \cdot B \cdot C$

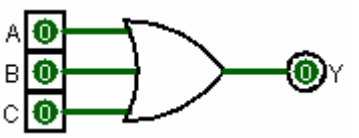
C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3. Nacrtajte dijagram izlaznoga napona sklopa I za zadanu pobudu na ulazu.



4. Navedite logička svojstva sklopa ILI.

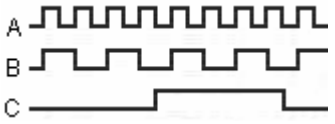
5. Nacrtajte simbol sklopa ILI s tri ulaza i napišite pripadni algebarski izraz i tablicu stanja.



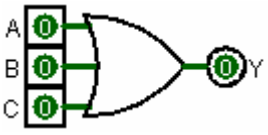
$Y = A + B + C$

C	B	A	Y
0	0	0	0
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

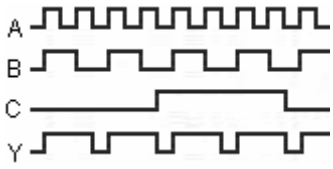
6. Nacrtajte dijagram izlaznoga napona sklopa ILI za zadanu pobudu na ulazu.



(a)



(b)

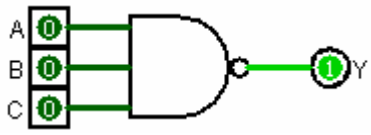


(c)

7. Objasnite logička svojstva sklopa NE.

8. Objasnite logička svojstva sklopa NI.

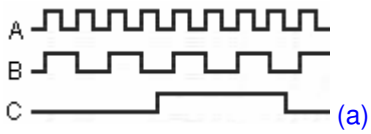
9. Nacrtajte simbol sklopa NI s tri ulaza i napišite pripadni algebarski izraz i tablicu stanja.



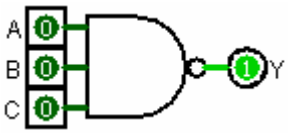
$Y = \overline{A \cdot B \cdot C}$

C	B	A	Y
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	0

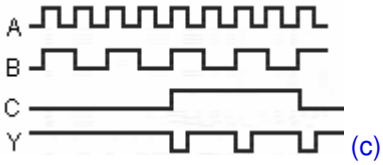
10. Nacrtajte dijagram izlaznoga napona sklopa NI uz zadanu pobudu na ulazu.



(a)



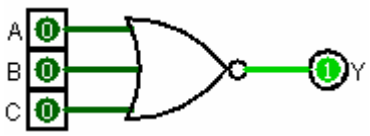
(b)



(c)

11. Objasnite logička svojstva sklopa NILI.

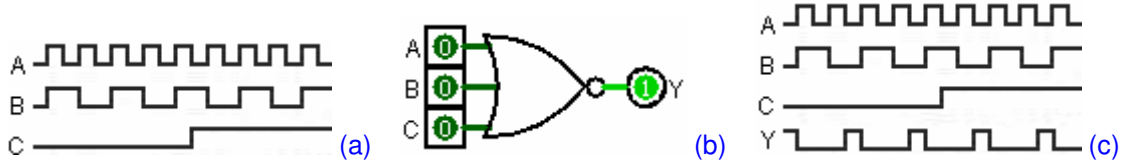
12. Nacrtajte simbol sklopa NILI s tri ulaza te napišite pripadni algebarski izraz i tablicu stanja.



$Y = \overline{A + B + C}$

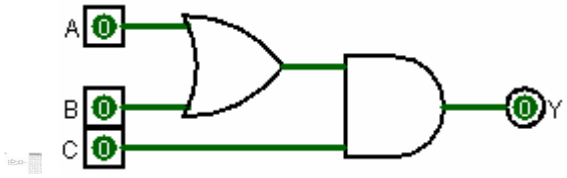
C	B	A	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

13. Nacrtajte dijagram izlaznoga napona sklopa NILI za zadanu pobudu.



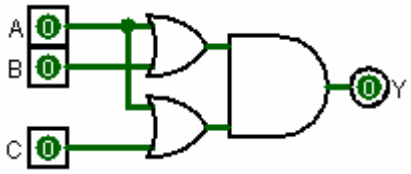
14. Nacrtajte logičku shemu sklopa koji obavlja logičku operaciju: Y = (A + B) · C. Tablicom stanja prikažite logička svojstva sklopa.

Y = (A + B) · C



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

15. Za sklop prema zadanoj logičkoj shemi napišite algebarski izraz i tablicu stanja.



Y = (A + B) (A + C).

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

2.2. 2.2. LOGIČKA ALGEBRA

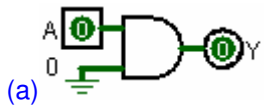
Engleski matematičar George Boole objavio je 1854. knjigu u kojoj je utvrdio postavke matematičkih operacija koje su se naknadno pokazale kao izuzetno pogodno sredstvo za opis djelovanja elemenata, čija stanja mogu poprimiti samo dvije vrijednosti, a to je slučaj s digitalnim sklopovima.

U ovome poglavlju razmotrit će se osnovna pravila, teoremi i zakoni logičke, odnosno Booleove algebre te njihova primjena pri realizaciji složenih logičkih sklopova.

2.2.1. 2.2.1. TEOREMI[6] LOGIČKE ALGEBRE

Pravila za operacije s jednom ulaznom *promjenjivom veličinom* - *varijablom* (*variable*) prikazuje niz slīka. Ulazna veličina A može imati vrijednost 0 ili 1. Osnovna pravila logičke algebre prikazuje niz sljedećih 9 simulacijskih prikaza (LogiSim), logičkih izraza i tablica istine:

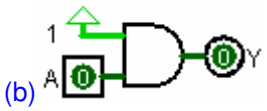
1. Ako je na jednome ulazu sklopa I varijabla A, a drugi ulaz je u stanju 0, izlaz Y stalno je u stanju 0 bez obzira na vrijednost varijable A.



A	0	Y
0	0	0
1	0	0

A · 0 = 0

2. Ako je jedan od ulaza sklopa I u stanju 1, onda je izlaz sklopa uvijek jednak ulaznoj veličini A.



A	1	Y
0	1	0
1	1	1

A · 1 = A

[6] Osnovno svojstvo teorema je oblikovanje rečenice prema pravilu *ako-onda* (Primjer: "Ako su ulazi I sklopa u logičkoj jedinici, onda je i njegov izlaz u logičkoj jedinici").

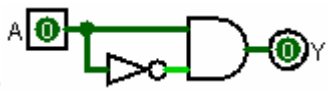
3. Ako se na oba ulaza sklopa I dovede ista varijabla, izlaz je jednak toj varijabli.

(c)

A	A	Y
0	0	0
1	1	1

$A \cdot A = A$

4. Ako se na jedan ulaz sklopa I, dovede varijabla A, a na drugi ulaz njezin komplement, izlaz je uvijek u stanju 0 bez obzira na vrijednost varijable, jer je jedan od ulaza sklopa I uvijek u stanju 0.



(d)

A	$\overline{A}$	Y
0	1	0
1	0	0

$A \cdot \overline{A} = 0$

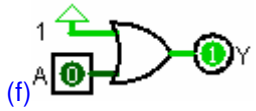
5. Ako je na jednome ulazu sklopa ILI varijabla A, a na drugome ulazu je stanje 0, tada je izlaz Y uvijek jednak ulaznoj veličini A.

(e)

A	0	Y
0	0	0
1	0	1

$A + 0 = A$

6. Ako je drugi ulaz sklopa ILI u stanju 1, izlaz Y je stalno u stanju 1 bez obzira na vrijednost ulazne varijable A.



(f)

A	1	Y
0	1	1
1	1	1

$A + 1 = 1$

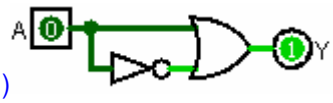
7. Ako se na oba ulaza sklopa ILI dovedu iste ulazne veličine, izlaz je jednak stanju ulaza.

(g)

A	A	Y
0	0	0
1	1	1

$A + A = A$

8. Ako se na jedan ulaz sklopa ILI dovede varijabla A, a na drugi ulaz njezin komplement A, izlaz je uvijek u stanju 1 bez obzira na vrijednost varijable A, jer je jedan od ulaza uvijek u stanju 1.

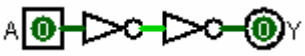


(h)

A	$\overline{A}$	Y
0	1	1
1	0	1

$A + \overline{A} = 1$

9. Ako se ulazna varijabla A invertira dva puta uzastopce, rezultat je jednak vrijednosti same ulazne veličine A.



(i)

A	$\overline{A}$	$\overline{\overline{A}}$	Y
0	1	0	0
1	0	1	1

$\overline{\overline{A}} = A$

Prethodno oslikana pravila mogu se prikazati u obliku tablice:

Tablica 2.x Osnovna pravila kao temelj minimizacije logičkih sklopova.

$0 \cdot 0 = 0$	$0 + 0 = 0$	$A \cdot 0 = 0$	$A + 0 = A$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$A \cdot 1 = A$	$A + 1 = 1$
$1 \cdot 1 = 1$	$1 + 1 = 1$	$A \cdot A = A$	$A + A = A$
$\sim 0 = 1$	$\sim 1 = 0$	$A \cdot \sim A = 0$	$A + \sim A = 1$
		$\sim\sim A = A$	

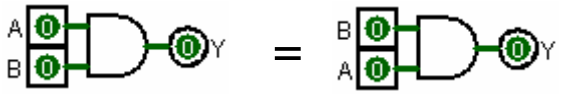
2.2.2. 2.2.2. ZAKONI LOGIČKE ALGEBRE

Pravila za operacije s više ulaznih varijabli nazivaju se *zakoni logičke algebre*. To su zakoni *komutacije*, *asocijacije* i *distribucije*.

Redoslijed dovođenja ulaznih varijabli na ulaz logičkoga sklopa nema utjecaja na rezultat logičke operacije. To pravilo zove se *zakon komutacije* (*commutative law*).

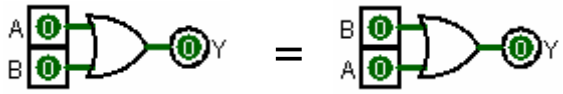
Zakon komutacije za logičko množenje dviju varijabli

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



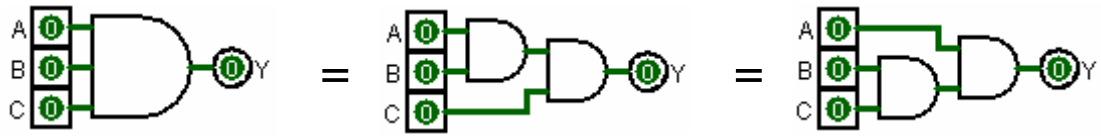
Zakon komutacije za logičko zbrajanje dviju varijabli

$$A + B = B + A$$



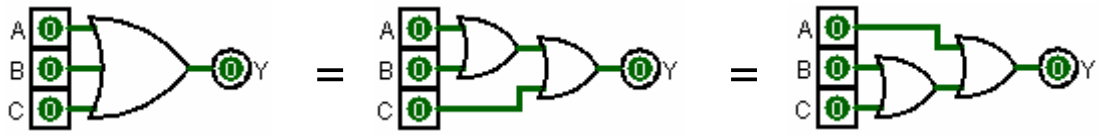
$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Zakoni asocijacije za logičko množenje triju varijabli



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

Zakoni asocijacije za logičko zbrajanje triju varijabli



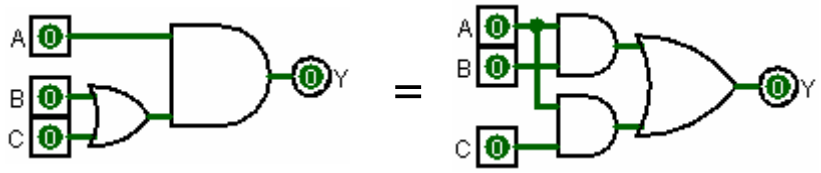
Zakoni komutacije i asocijacije su očigledni, dok je za zakone distribucije, dokaz izveden tablicom stanja.

Način nizanja ulaznih veličina kod I i ILI operacije, a obzirom na zakone asocijacije (*associative laws*) ne utječe na konačan rezultat.

Ako u logičkome zbroju dvaju ili više članova postoji zajednički član, prema prvome zakonu distribucije (*distributive law*), taj član može se izlučiti ispred zagrade.

(a) Prvi zakon distribucije

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

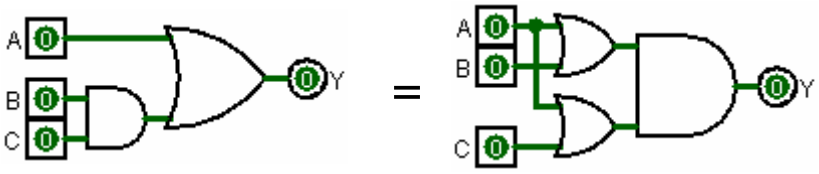


C	B	A	(B + C)	(B + C) · A	Y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	1

C	B	A	(A · B)	(A · C)	Y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	1	1	1

(b) Drugi zakon distribucije

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



C	B	A	(B · C)	(B · C) + A = Y
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

C	B	A	(A + B)	(A + C)	Y
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Drugi zakon distribucije pokazuje jednakost logičkoga zbroja jedne varijable (A) i umnoška dviju varijabli (B C) s logičkim umnoškom dvaju zbrojeva varijable A posebno sa svakim od članova B i C.

2.2.2.1. Pojednostavnjenje logičke operacije primjenom pravila logičke algebre

43. Primjer: Pojednostavnite logičku funkciju Y = A (A + B)

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

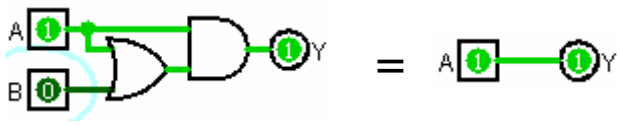
$= (A + 0)\cdot(A + B)$ pravilo $A + 0 = A$

$= A + B \cdot 0$ zakon distribucije, pravilo $B \cdot 0 = 0$

$= A + 0$ pravilo $A + 0 = A$

$= A$

Izlaz Y ne ovisi o B pa nije potreban nikakav sklop.



44. Primjer: Pojednostavnite logičku, operaciju Y = A + (A · B)

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

$= (A + A)\cdot(A + B)$ zakon distribucije, pravilo $A + A = A$

$= A \cdot (A + B)$

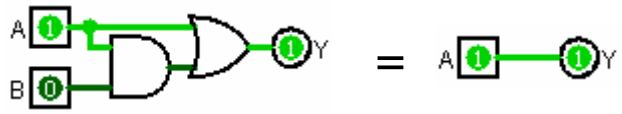
$= (A + 0)\cdot(A + B)$ pravilo $A + 0 = A$, zakon distribucije

$= A + (B \cdot 0)$ pravilo $B \cdot 0 = 0$

$= A + 0$ pravilo $A + 0 = A$

$= A$

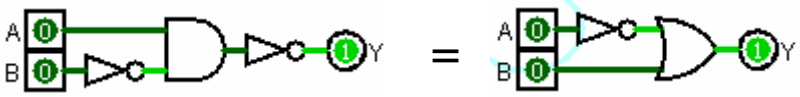
Izlaz Y ne ovisi o B pa nije potreban nikakav sklop.



Ovih nekoliko jednostavnih primjera pokazuje da se primjenom osnovnih zakona logičke algebre mogu znatno pojednostavniti složene logičke funkcije. Ponekad se događa da je izlaz jednostavno jednak nekoj od ulaznih varijabli, a uopće ne ovisi da o drugim varijablama (primjer 43 i primjer 44). Za sklopove koji nemaju svoj minimalan oblik kažemo da imaju zalihost (*redundancy*).

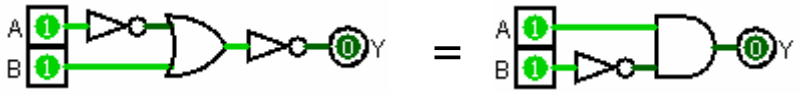
U sklopovima sa zalihošću može se dogoditi da je neki od sklopova u kvaru, a izlaz je i dalje ispravan. Zbog pojednostavnjenja sklopa često je prikladno NE operaciju, koja se nalazi iza sklopa I ili ILI, prebaciti u ulazni krug tih sklopova, ili obrnuto. Što se pri tome događa pokazano je na slici.

Prebacivanje operacije NE s izlaza na ulaz



$\overline{A \cdot B} = \overline{A} + \overline{B} = \overline{A} + B$ (De Morganov teorem)

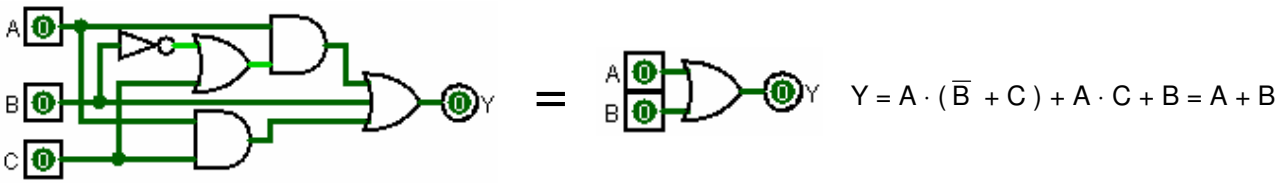
Prebacivanje operacije NE s izlaza na ulaz



$\overline{\overline{A} + B} = \overline{A} \cdot \overline{B} = A \cdot \overline{B}$ (De Morganov teorem)

Vidi se da "provlačenjem" operacije NE kroz sklop zahtijeva promjenu položaja negacije na ulazima i izmjenu logičke funkcije I u ILI, odnosno obratno. Za dokaz može poslužiti De Morganov teorem (vidi u nastavku).

45. Primjer: Pojednostavnjene logičke operacije primjenom pravila logičke algebre



Primjer 45 pokazuje kako se primjenom pravila logičke algebre može pojednostavniti složena logička operacija, odnosno složen logički sklop koji tu operaciju obavlja. Postupak pojednostavnjenja logičkoga sklopa naziva se *minimizacija*. Sklop zadan algebarskim izrazom:

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

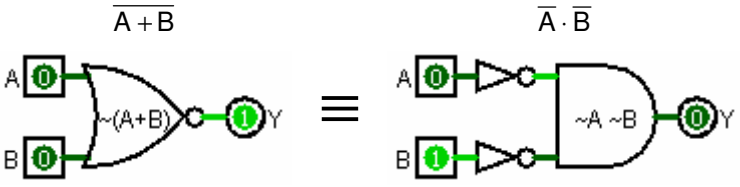
nije u svome minimalnome obliku. Provedenim postupkom minimizacije svodi se na minimalan oblik Y = A + B. Za sklopove koji nemaju svoj minimalan oblik kaže se da imaju *zalihost*. Sklop sa zalihošću složeniji je, ali ima veću pouzdanost. Neki dio složenoga sklopa može se pokvariti, a izlazi ipak daju ispravan rezultat.

2.2.3. 2.2.3. DE MORGANOVI TEOREMI

Među najvažnijim teoremima logičke algebre su *De Morganovi teoremi*. Upotrebljavaju se vrlo često u postupku pojednostavnjenja logičkih operacija logičkih umnožaka ili logičkih zbrojeva s invertiranim ulaznim veličinama.

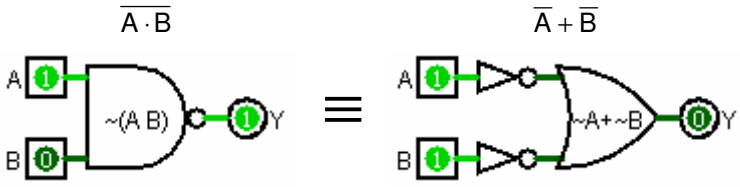
De Morganov teorem 1

A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0



De Morganov teorem 2

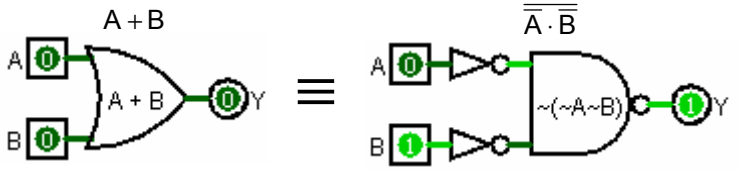
A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	1
1	0	0	1	1
0	1	1	0	1
1	1	0	0	0



De Morganovi teoremi pokazuju međusobnu *dvojnost* (*duality*) logičkih operacija i realizaciju logičkih sklopova samo sklopovima NI i NILI.

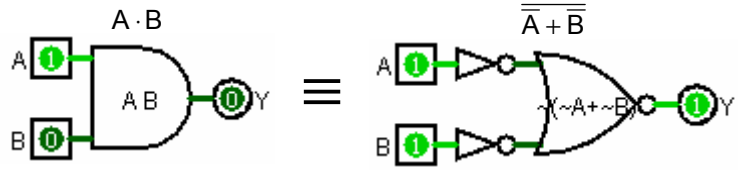
Drugi oblik De Morganova teorema 1

A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1



Drugi oblik De Morganova teorema 2

A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1



46. Primjer: Primjenom De Morganovih teorema pojednostavnite logičku operaciju

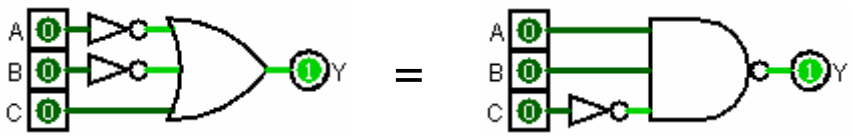
$$Y = \overline{A} + \overline{B} + C$$

$$Y = \overline{A} + \overline{B} + C = \overline{\overline{\overline{A}} \cdot \overline{\overline{B}}} + C = \overline{\overline{\overline{A} \cdot \overline{\overline{B}} \cdot \overline{C}}} = \overline{A \cdot B \cdot \overline{C}}$$

Primjenom De Morganova teorema smanjio se broj inverzija ulaznih veličina.

Primjena De Morganovih teorema

(a) zadan sklop (b) pojednostavnjen sklop



47. Primjer: Pojednostavnite logičku, operaciju: $Y = A + \overline{A} \cdot B$

$Y = A + \overline{A} \cdot B =$

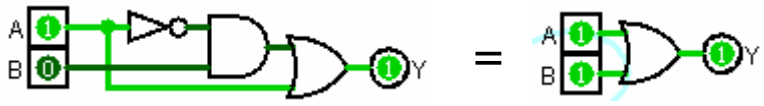
$= \overline{\overline{A + \overline{A} \cdot B}}$ De Morganov teorem

$= \overline{\overline{A} \cdot (A + \overline{B})}$ De Morganov teorem

$= \overline{\overline{A} \cdot A + \overline{A} \cdot \overline{B}}$ pravilo $A \cdot \overline{A} = 0$

$= \overline{\overline{A}} + \overline{\overline{B}}$ De Morganov teorem, pravilo $\overline{\overline{A}} = A$

$= A + B$



48. Primjer: Pojednostavnite logičku, operaciju $Y = A \cdot (\overline{B} + C) + A \cdot C + B$:

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

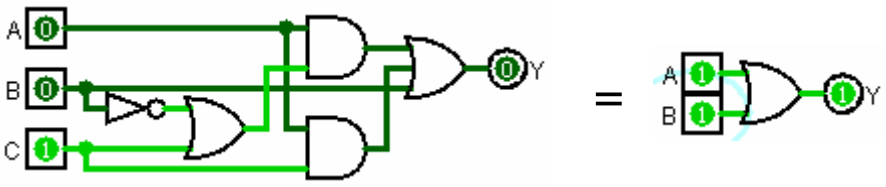
$= A \cdot \overline{B} + A \cdot C + A \cdot C + B$ zakon distribucije, pravilo $A + A = A$

$= A \cdot \overline{B} + A \cdot C + B$ zakon komutacije

$= B + A + A \cdot C$ zakon asocijacije, De Morganov teorem

$= B + A(1 + C)$ zakon asocijacije

$= B + A$

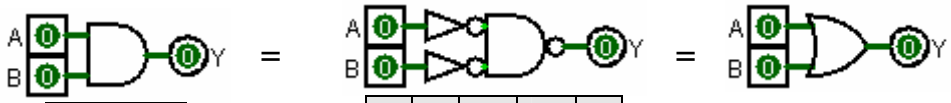


2.2.4. 2.2.4. DVOJNOST LOGIČKIH OPERACIJA

Ako se invertiranjem ulaznih i izlaznih veličina, iz jedne logičke operacije dobije druga i obrnuto, onda su te dvije operacije međusobno *dvojne* (*dualne*). Sljedeće slike prikazuje međusobnu dvojnost logičkih operacija I i ILI.

Međusobna dvojnost logičkih operacija I i ILI

(a) operacija I (b) operacija ILI



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

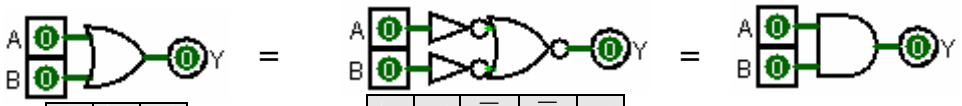
$Y = A \cdot B$

B	A	$\overline{B}$	$\overline{A}$	Y
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

$Y = \overline{\overline{A} \cdot \overline{B}} = A + B$

Međusobna dvojnost logičkih operacija ILI i I

(a) operacija ILI (b) operacija I



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

$Y = A + B$

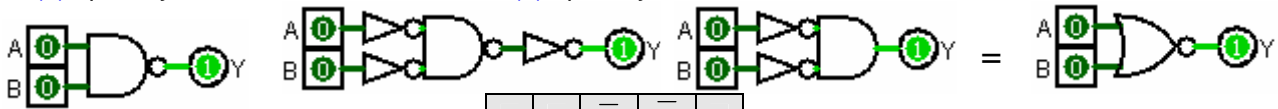
B	A	$\overline{B}$	$\overline{A}$	Y
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

$Y = \overline{\overline{A} + \overline{B}} = A \cdot B$

To znači da se pomoću logičkih sklopova I i NE može realizirati operacija ILI, odnosno pomoću sklopova ILI i NE operacija I. Prema tome, pri korištenju osnovnih logičkih operacija može se jedna od međusobno dvojnih operacija izostaviti, jer se može realizirati uporabom preostalih dviju osnovnih operacija. Uporabom samo sklopa I i NE, odnosno ILI i NE mogu se obaviti sve jednostavne i složene logičke operacije.

Međusobna dvojnost logičkih operacija NI i NILI

(a) operacija NI (b) operacija NILI



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

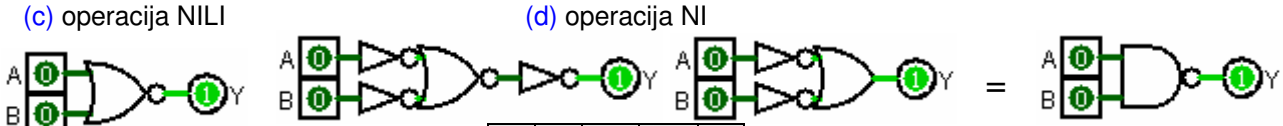
$Y = \overline{A \cdot B}$

A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	1
1	0	0	0	0
0	1	1	1	0
1	1	0	0	0

$Y = \overline{\overline{A} \cdot \overline{B}} = \overline{A} \cdot \overline{B} = \overline{A + B}$

Međusobna dvojnost logičkih operacija NILI i NI

(c) operacija NILI (d) operacija NI



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

$Y = \overline{A + B}$

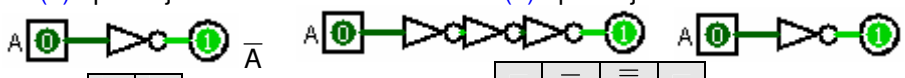
A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	1
1	0	0	1	1
0	1	1	0	1
1	1	0	0	0

$Y = \overline{\overline{A} + \overline{B}} = \overline{A} + \overline{B} = \overline{A \cdot B}$

Operacija NE nema sebi dvojne operacije. Invertiranjem ulaznih i izlaznih veličina NE operacije ponovo se dobije operacija NE.

Dvojnost operacije NE

(a) operacija NE (b) operacija NE



A	Y
0	1
1	0

$Y = \overline{A}$

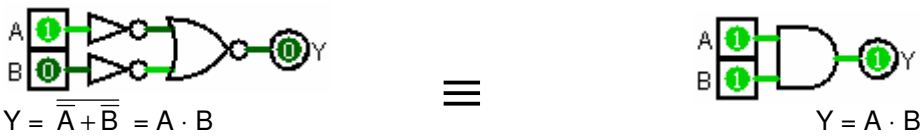
A	$\overline{A}$	$\overline{\overline{A}}$	Y
0	1	0	1
1	0	1	0

$Y = \overline{A}$

Ako je na ulazu ili izlazu logičkoga sklopa znak inverzije (kružnica), kaže se da na ulazu, odnosno izlazu, aktivan signal ima logičko stanje 0 (*active low*). Ako na ulazu i izlazu sklopa nema znaka inverzije, tada na njima aktivan signal ima logičko stanje 1 (*active high*). To znači da je operacija NE sama sebi dvojna.

Predstavljanje dvojnosti simbola logičkih sklopova NILI i I

$$\overline{\overline{A}+\overline{B}} = A \cdot B$$

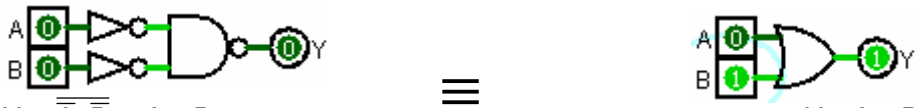


$Y = \overline{\overline{A}+\overline{B}} = A \cdot B$ $\equiv$ $Y = A \cdot B$

Izlaz je 0 ako je bilo koji ulaz 0. (a) Izlaz je 1 ako su svi ulazi 1. (b)

Predstavljanje dvojnosti simbola logičkih sklopova NI i ILI

$$\overline{\overline{A}\cdot\overline{B}} = A + B$$



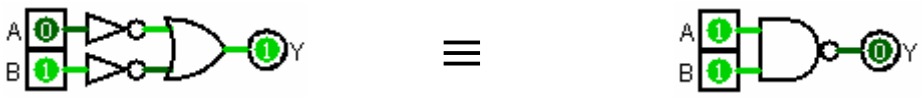
$Y = \overline{\overline{A}\cdot\overline{B}} = A + B$ $\equiv$ $Y = A + B$

Izlaz je 0 ako su svi ulazi 0. (c) Izlaz je 1 ako je bilo koji ulaz 1. (d)

Za sklop I prikazan standardnim simbolom može se iskazati da aktivan signal na izlazu ima logičko stanje 1, tj. izlaz je u stanju 1 ako su svi ulazi u stanju 1. Sklop I prikazan dvojnim simbolom na izlazu ima aktivan signal 0, tj. izlaz je u stanju 0 ako je bilo koji od ulaza u stanju 0. Na isti način moguće su predodžbe i ostalih osnovnih logičkih sklopova.

Predstavljanje dvojnosti simbola logičkih sklopova NILI i NI

$$\overline{A}+\overline{B} = \overline{A\cdot B}$$

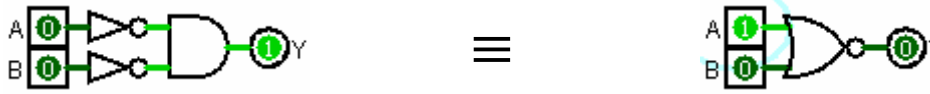


$Y = \overline{A}+\overline{B} = \overline{A\cdot B}$ $\equiv$ $Y = \overline{A\cdot B}$

Izlaz je 1 ako je bilo koji ulaz 0. (a) Izlaz je 0 ako su svi ulazi 1. (b)

Predstavljanje dvojnosti simbola logičkih sklopova NI i NILI

$$\overline{A}\cdot\overline{B} = \overline{A+B}$$



$Y = \overline{A}\cdot\overline{B} = \overline{A+B}$ $\equiv$ $Y = \overline{A+B}$

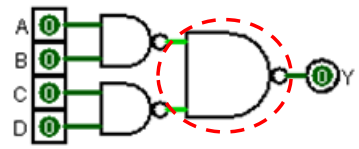
Izlaz je 1 ako su svi ulazi 0. (c) Izlaz je 0 ako je bilo koji ulaz 1. (d)

Korištenje dvojnih simbola u logičkim shemama složenih logičkih sklopova omogućuje lakšu predodžbu svojstava i rada sklopa.

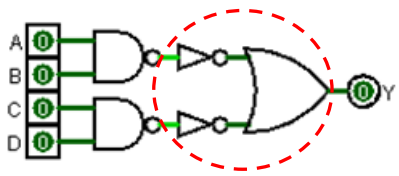
49. Primjer: Primjena dvojnih simbola za preoblikovanje logičke sheme sklopa

Primjenom dvojnih simbola treba izvesti sklop pomoću sklopova NI, u shemu izvedbe s osnovnim logičkim sklopovima.

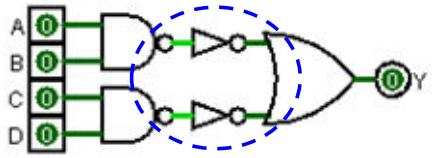
(a)



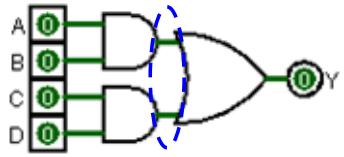
(b)



(c)



(d)



(e)

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

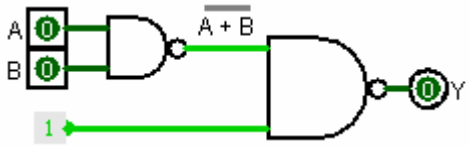
U logičkoj shemi složenoga sklopa izvedenoga pomoću tri sklopa NI moguće je zamijeniti sklopove NI dvojnim simbolima. Ako se izlazni sklop NI ($\overline{A \cdot B}$), zamijeni dvojnim sklopom $\overline{A \cdot B} = \overline{A} + \overline{B}$ i odstrane se uzastopne inverzije, dobije se logička shema s dva I sklopa i jednim ILI sklopom. Aktivan signal na ulazima i izlazu je 1. Iz ove logičke sheme lako je uočljivo svojstvo sklopa po kojemu je izlaz 1 ako su A = B = 1 ili C = D = 1. Ovakav prikaz povoljan je, ako sklop stanjem 1 na izlazu, aktivira sljedeći sklop ili uređaj.

Opće pravilo za prikaz logičkih shema složenih logičkih sklopova kaže da se, kada god je to moguće, međusobno vezuju invertirajući izlazi i ulazi (aktivan signal 0), odnosno neinvertirajući izlazi i ulazi (aktivan signal 1).

2.2.5. 2.2.5. UNIVERZALNOST LOGIČKIH SKLOPOVA NI I NILI

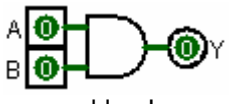
Svojstvo sklopova NI i NILI jest da se svaka osnovna, a prema tome i složena, logička operacija može ostvariti uporabom samo sklopova NI ili samo sklopova NILI.

(a) Izvedba logičkoga sklopa I uporabom sklopova NI



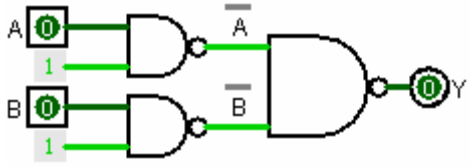
$Y = \overline{\overline{A \cdot B}} = A \cdot B$

A	B	$\overline{A \cdot B}$	Y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1



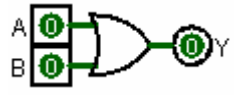
sklop I

(b) Izvedba logičkoga sklopa ILI uporabom sklopova NI



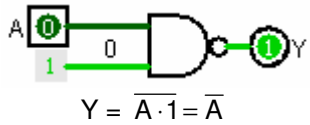
$Y = \overline{A} \cdot \overline{B} = A + B$

A	B	$\overline{A}$	$\overline{B}$	Y
0	0	1	1	0
1	0	0	1	1
0	1	1	0	1
1	1	0	0	1



sklop ILI

(c) Izvedba logičkoga sklopa NE uporabom sklopova NI



$Y = \overline{A \cdot 1} = \overline{A}$

1	A	Y
1	0	1
0	1	0

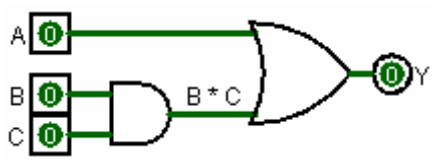
sklop NE

Ako se sklop NI upotrebljava kao sklop NE, koristi se samo jedan ulaz. Preostali ulazi moraju biti u stanju 1. Budući da sklop NI daje invertiranu operaciju I, ponovnim invertiranjem izlaza sklopa NI dobije se operacija I. Postupak za realizaciju sklopa ILI pomoću sklopa NI slijedi iz De Morganova teorema. Ulazne veličine prvo se invertiraju, a zatim dovedu na ulaz sklopa NI.

50. Primjer: Uporabom sklopova NI, nacrtati shemu sklopa za logičku operaciju Y = A + B · C.

(a) Osnovni logički sklopovi I i ILI

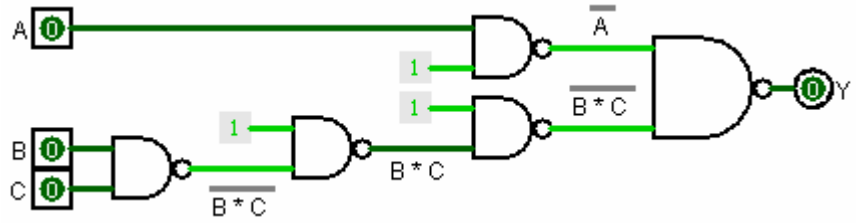
Y = A + B · C



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

U logičkoj shemi sklopa na prethodnoj slici, izvedene uporabom osnovnih logičkih sklopova zamijene se osnovni logički sklopovi njihovim ekvivalentima izvedenima pomoću sklopova NI.

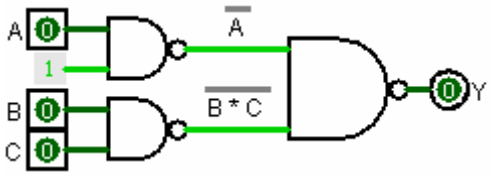
(b) Izvedba složene logičke operacije uporabom sklopova NI



$$Y = \overline{\overline{A} \cdot B \cdot C}$$

Tako dobivena shema složenoga sklopa izvedenoga pomoću sklopova NI može se pojednostavniti izostavljanjem dvaju uzastopnih invertora.

(c) Minimizacija

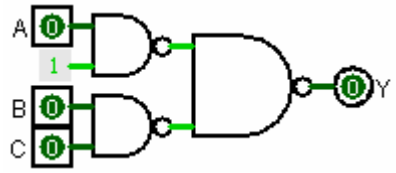


$$Y = \overline{\overline{A} \cdot \overline{B \cdot C}}$$

Do istoga rezultata moguće je doći izravnom primjenom De Morganova teorema na zadanu logičku operaciju. Na temelju dobivene logičke jednadžbe dolazi se do logičke sheme sklopa izvedenoga pomoću sklopova NI.

(d) Konačan rezultat izvedbe složene logičke operacije uporabom sklopova NI

$$Y = A + B \cdot C = \overline{\overline{A} \cdot \overline{B \cdot C}}$$

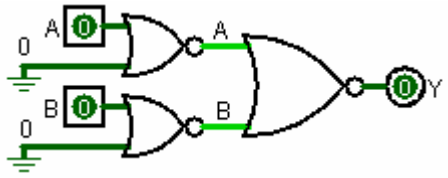


C	B	A	$\overline{A}$	$\overline{B \cdot C}$	Y
0	0	0	1	1	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	0	1

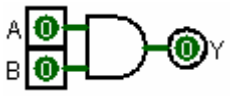
Pripadna tablica stanja pokazuje da sklop obavlja jednaku operaciju kao i sklop izveden uporabom osnovnih logičkih sklopova.

Ako se sklop NILI upotrebljava kao sklop NE koristi se jedan ulaz, a preostali ulazi moraju biti u stanju 0.

(a) Izvedba osnovnoga logičkoga sklopa I uporabom sklopova NILI



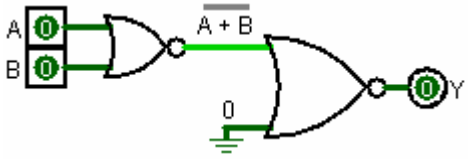
B	A	$\overline{B}$	$\overline{A}$	Y
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1



sklop I

$$Y = \overline{\overline{A} + \overline{B}} = A \cdot B$$

(b) Izvedba osnovnoga logičkoga sklopa ILI uporabom sklopova NILI



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

sklop ILI

$$Y = \overline{\overline{A+B}} = A + B$$

(c) Izvedba osnovnoga logičkoga sklopa NE uporabom sklopova NILI

1	A	Y
1	0	1
0	1	0

sklop NE

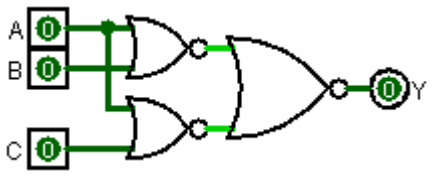
$$Y = \overline{A+0} = \overline{A}$$

Kako sklop NILI daje invertiranu operaciju ILI, ponovnim invertiranjem izlaza sklopa NILI opet se dobije operacija ILI. Postupak za izvedbu sklopa I uporabom sklopova NILI slijedi iz De Morganova teorema. Ulazne veličine najprije se invertiraju, a onda dovedu na ulaz sklopa NILI.

51. Primjer: Nacrtati shemu sklopa izvedenog samo uporabom sklopova NILI

za logičku operaciju: $Y = (A + B) \cdot (A + D)$.

$$Y = (A + B) \cdot (A + C) = \overline{\overline{A+B} + \overline{A+C}}$$ (a)



(b)

2.2.6. 2.2.6. PREGLED KLJUČNIH POJMOVA

Booleova algebra (*Boolean algebra*)

- logička algebra, nazvana prema Georgeu Booleu koji je utvrdio osnovne postavke logičke algebre

De Morganovi teoremi (*De Morgan Theorems*)

$$\overline{X+Y} = \overline{X} \cdot \overline{Y} \text{ odnosno } X + Y = \overline{\overline{X} \cdot \overline{Y}}$$

i

$$\overline{X \cdot Y} = \overline{X} + \overline{Y} \text{ odnosno } X \cdot Y = \overline{\overline{X} + \overline{Y}}$$

dvojnost (dualnost) logičkih operacija

- međusobno su dvojne one logičke operacije kôd kojih se invertiranjem ulaznih i izlaznih veličina od jedne operacije dobije druga; međusobno su dvojne operacije I i ILI, odnosno NI i NILI

logička algebra (Booleova algebra)

- matematičke operacije s varijablama koje mogu imati samo dvije vrijednosti

minimalan oblik logičke operacije

- složena logička operacija izvedena najmanjim mogućim brojem osnovnih logičkih operacija

osnovna pravila logičke algebre za operacije jednom varijablom

$$A \cdot 0 = 0 \quad A \cdot 1 = A \qquad A \cdot A = A \qquad \overline{A} \cdot A = 0 \qquad \overline{\overline{A}} = A$$

$$A + 0 = A \; A + 1 = 1 \qquad A + A = A \qquad \overline{A} + A = 1$$

univerzalnost logičkih sklopova

- svojstvo sklopova NI i NILI da se svaka osnovna i složena logička operacija može izvesti samo sklopovima NI, odnosno samo sklopovima NILI

zakoni asocijacije (*associative laws*)

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

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

zakoni distribucije (*distributive laws*)

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

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

zakoni komutacije (*commutative laws*)

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

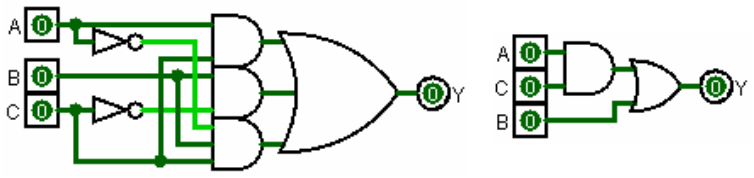
$$A + B = B + A$$

2.2.7. 2.2.7. PITANJA I ZADACI ZA PONAVLJANJE

1. Navedite sva osnovna pravila logičke algebre za operacije s jednom ulaznom veličinom.

2. Navedite zakone komutacije, asocijacije i distribucije. Za svaki zakon navedite pripadne sheme logičkih sklopova kojima se može dokazati valjanost zakona.

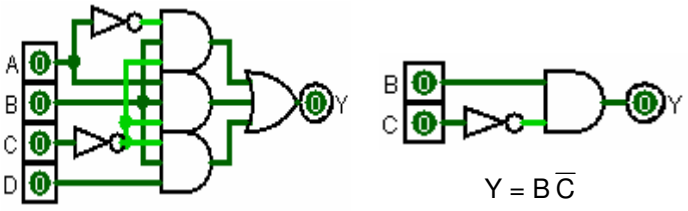
3. Primjenom pravila logičke algebre pojednostavnite logičku operaciju: Y = AC + B~C + ~ABC. Nacrtajte logičku shemu sklopa za zadanu i pojednostavnjenu operaciju. Rezultat provjerite tablicama stanja za oba sklopa.



Y = AC + B

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

4. Primjenom pravila logičke algebre pojednostavnite logičku operaciju: Y = ~A B~C + A B~C + B~C D. Nacrtajte logičku shemu sklopa za zadanu i pojednostavnjenu operaciju. Rezultate provjerite tablicama stanja za oba sklopa.



$Y = B\overline{C}$

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

5. Navedite De Morganove teoreme i pripadne logičke sklopove kojima je moguće dokazati valjanost teorema.

6. Primjenom De Morganovih teorema u logičkome izrazu za operaciju: $Y = (\overline{A} + C) \cdot (B + \overline{D})$. Zamijenite sve znakove · (operacija I) znakovima + (operacija ILI).

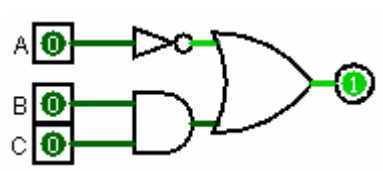
7. Primjenom De Morganovih teorema u logičkome izrazu za operaciju: $\overline{A} + B \cdot C$, zamijenite sve znakove + (operacija ILI) znakovima · (operacija I).

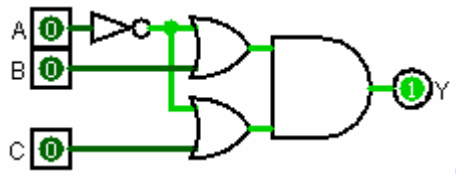
8. Što znači dvojnost (dualnost) logičkih operacija? Koje su logičke operacije međusobno dvojne?

9. Kako se mogu ostvariti osnovni logički sklopovi uporabom sklopova NI?

10. Kako se mogu ostvariti osnovni logički sklopovi uporabom sklopova NILI?

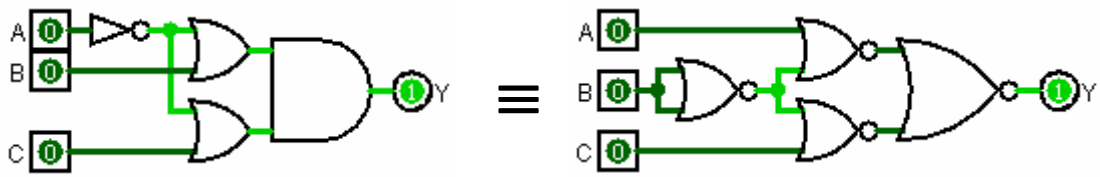
11. Izvedite uporabom sklopova NI zadani logički sklop.



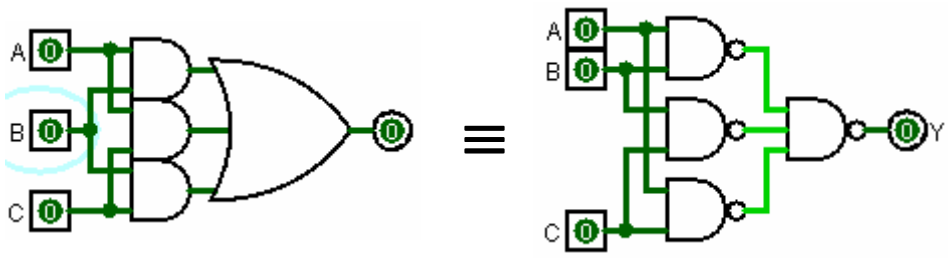
(a)


(b)

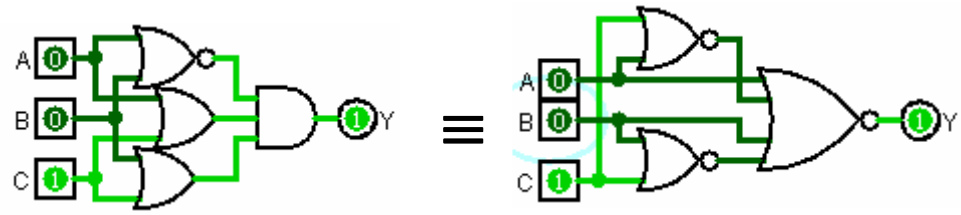
12. Izvedite uporabom sklopova NILI zadani logički sklop.



13. Izvedite uporabom sklopova NI, sklop koji obavlja logičku operaciju: $Y = A\,B + A\,C + B\,C = \overline{\overline{A \cdot B} + \overline{A \cdot C} + \overline{B \cdot C}}$



14. Izvedite uporabom sklopova NILI, sklop koji obavlja logičku operaciju: $Y = \overline{(A + B)} \cdot (A + C) \cdot (B + C)$.



~(A + B)(A + C)(B + C)

2.3. 2.3. SLOŽENI LOGIČKI SKLOPOVI

U ovome poglavlju razmotrit će se projektiranje složenih logičkih sklopova. Dvije osnovne mogućnosti projektiranja složenih logičkih sklopova su:

- metoda logičkoga zbroja logičkih umnožaka (*sum-of-products method*) ili zbroj *minterma*, te
- metoda logičkoga umnoška logičkih zbrojeva (*product-of-sums method*) ili umnožak *Maksterma*.

Dobiveni algebarski izrazi za složene logičke operacije nisu uvijek minimalnoga oblika pa se provodi *minimizacija*.

2.3.1. 2.3.1. MINTERM

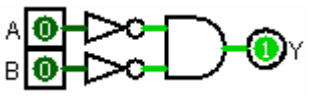
S dvije ulazne varijable moguće je napraviti četiri različite operacije logičkih umnožaka (*fundamental products*). To su:

$$\overline{A}\cdot\overline{B},\ A\cdot\overline{B},\ \overline{A}\cdot B \text{ i } A\cdot B.$$

Ako se za svaku od ovih operacija napravi tablica stanja za moguće vrijednosti varijabli na ulazu, vidi se da se u izlaznome stupcu svaki put dobije *samo jedna jedinica*.

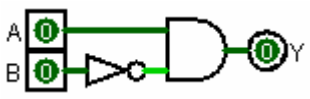
Mogući logički umnošci s dvije ulazne varijable

$Y = \overline{A}\cdot\overline{B} = m_0$



A	B	m_0
0	0	1
1	0	0
0	1	0
1	1	0

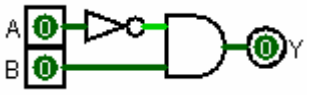
$Y = A\cdot\overline{B} = m_1$



A	B	m_1
0	0	0
1	0	1
0	1	0
1	1	0

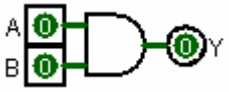
Mogući logički umnošci s dvije ulazne varijable

$Y = \overline{A}\cdot B = m_2$



A	B	m_2
0	0	0
1	0	0
0	1	1
1	1	0

$Y = A\cdot B = m_3$



A	B	m_3
0	0	0
1	0	0
0	1	0
1	1	1

Ovakva logička operacija, koja na izlazu daje jedinicu samo za jednu ulaznu kombinaciju, a za sve ostale ulazne kombinacije na izlazu je logička nula, naziva se *minterm*, jer je broj jedinica na izlazu minimalan.

Funkcija *minterm* može se ostvariti tako da se koristi operacija I, time da se one ulazne veličine koje su u stanju 0 u kombinaciji što daje izlaz 1 prethodno invertiraju.

S dvije ulazne varijable moguće je izvesti četiri različita minterma, tj. upravo onoliko koliki je broj različitih kombinacija s dvije ulazne varijable, odnosno koliki je moguć broj osnovnih logičkih umnožaka. To znači da s tri ulazne varijable postoji osam minterma, s četiri šesnaest itd.

52. Primjer: Realizacija minterma m_2 s tri ulazne varijable.

Oznaka minterm m_2 znači da je izlaz jednak 1 za ulaznu kombinaciju 010. Stoga je logička jednadžba minterma:

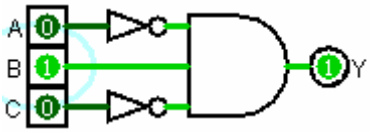
$$m_2 = \overline{A}\cdot\overline{B}\cdot C$$

što znači da ulazne veličine A i C treba prethodno invertirati i tada zajedno s B dovesti na ulaz I sklopa.

Tablica stanja i logička shema za minterm m_2 s tri ulazne varijable

C	B	A	Y	minterm
0	0	0	0	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	0	m_4
1	0	1	0	m_5
1	1	0	0	m_6
1	1	1	0	m_7

(a)



(b)

Ako je potrebno ostvariti logičku operaciju koja na izlazu daje 1 za više ulaznih kombinacija, koristi se logički zbroj minterma (*sum-of-products method*). Logička shema takvoga sklopa sastoji se od određenoga broja I sklopova s potrebnim invertorima i jednoga sklopa ILI na čije ulaze vežu se izlazi sklopova I (*AND-OR network*).

53. Primjer: Realizacija složene logičke operacije zadane tablicom stanja:

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

Iz tablice stanja proizlazi da izlaz Y mora biti u stanju 1 za dvije ulazne kombinacije: 010 i 101. Operaciju je moguće realizirati logičkim zbrojem dvaju minterma prema logičkoj jednadžbi

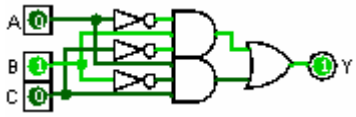
$$Y = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C$$

Za realizaciju ove operacije potrebna su dakle tri invertora, dva sklopa I s tri ulaza i jedan sklopa ILI s dva ulaza.

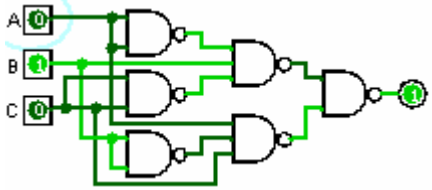
Realizacija složene logičke operacije pomoću logičkoga zbroja minterma

$$Y = \overline{\overline{A} \cdot B \cdot \overline{C} \cdot \overline{A \cdot \overline{B} \cdot C}}$$

a) izvedba osnovnim logičkim sklopovima



b) izvedba sklopovima NI



Primjenom De Morganova teorema na algebarski izraz, za zadanu funkciju dobije se izraz, što omogućuje realizaciju iste logičke operacije samo logičkim sklopovima NI. Iz logičke sheme vidi se da su za realizaciju iste operacije sada potrebna četiri NI sklopa s dva ulaza i dva NI sklopa s tri ulaza.

2.3.2. 2.3.2. MAKSTERM

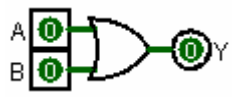
S dvije ulazne varijable moguće je učiniti četiri različite operacije logičkih zbrojeva. To su:

$$A+B,\ \overline{A}+B,\ A+\overline{B} \text{ i } \overline{A}+\overline{B}.$$

Ako se za svaku od ovih operacija napravi tablica stanja za moguće vrijednosti varijabli na ulazu, onda se vidi da se u izlaznome stupcu svaki put dobije samo jedna nula.

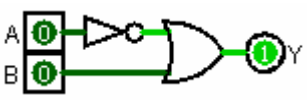
Mogući logički umnošci s dvije ulazne varijable

$Y=A+B=M_0$ $Y = A + B = M_0$



A	B	M_0
0	0	0
1	0	1
0	1	1
1	1	1

$Y = \overline{A} + B = M_1$



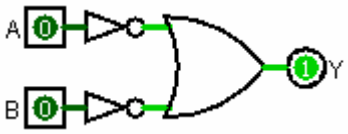
A	B	M_1
0	0	1
1	0	0
0	1	1
1	1	1

Mogući logički umnošci s dvije ulazne varijable

$Y = A + \overline{B} = M_2$

A	B	M_2
0	0	1
1	0	1
0	1	0
1	1	1

$Y = \overline{A} + \overline{B} = M_3$



A	B	M_3
0	0	1
1	0	1
0	1	1
1	1	0

Ovakva logička operacija koja na izlazu daje 0 samo za jednu ulaznu kombinaciju, a za sve ostale ulazne kombinacije na izlazu je jedinica, naziva se *Maksterm*, jer je broj jedinica na izlazu maksimalan.

Maksterm se može ostvariti tako da se koristi operacija ILI time da se one ulazne veličine koje su u stanju 1 u kombinaciji koja daje izlazno stanje 0, prethodno invertiraju.

S dvije ulazne varijable moguće je izvesti četiri različita Maksterma, tj. upravo onoliko koliki je broj različitih kombinacija s dvije ulazne varijable. Prema tome, s tri ulazne varijable postoji osam Maksterma, s četiri ulazne varijable šesnaest Maksterma, itd.

54. Primjer: Realizacija Maksterma M_2 s tri ulazne varijable.

Maksterm M_2 znači da je izlaz u stanju 0 za ulaznu kombinaciju 010. Iz toga slijedi algebarski izraz za M_2

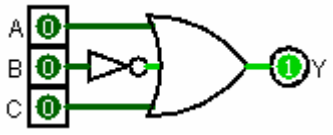
$$M_2 = A + \overline{B} + C$$

To znači da ulaznu veličinu B treba prethodno invertirati i zajedno s A i C dovesti na ulaz sklopa ILI.

Tablica stanja i logička shema za Maksterma M_2 s tri ulazne varijable

C	B	A	Y	maksterm
0	0	0	1	M_0
0	0	1	1	M_1
0	1	0	0	M_2
0	1	1	1	M_3
1	0	0	1	M_4
1	0	1	1	M_5
1	1	0	1	M_6
1	1	1	1	M_7

(a)



(b)

Ako je potrebno ostvariti logičku operaciju koja na izlazu daje stanje 0 za više ulaznih kombinacija, koristi se logički umnožak Maksterma, tj. umnoškom osnovnih logičkih zbrojeva (*the product-of-sums method*). Logička shema takvoga sklopa sastoji se od određenoga broja sklopova ILI s potrebnim invertorima i jednoga sklopa I na čije se ulaze vežu izlazi sklopova ILI (*OR-AND network*).

55. Primjer: Realizacija složene logičke operacije zadane tablicom stanja:

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

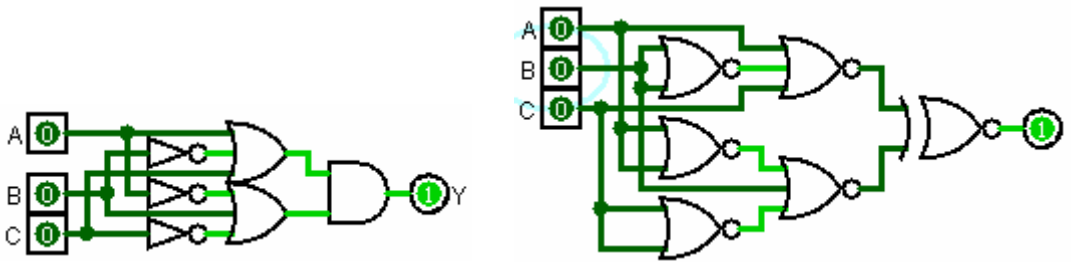
Iz tablice stanja proizlazi da izlaz mora biti u stanju 0 za dvije ulazne kombinacije. To su 010 i 101. To znači da se ova operacija može realizirati pomoću logičkoga umnoška dvaju Maksterma:

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

Dobiven algebarski izraz pokazuje da su za realizaciju, tablicom zadane operacije, potrebna tri invertora, dva sklopa ILI s tri ulaza te jedan sklop I s dva ulaza.

Realizacija složene logičke operacije pomoću logičkoga umnoška Maksterma

(a) izvedba osnovnim logičkim sklopovima (b) izvedba sklopovima NILI



Primjenom De Morganova teorema na dobiven algebarski izraz, dobije se oblik logičke jednadžbe koji omogućuju realizaciju iste logičke operacije samo sklopovima NILI:

$$Y = \overline{\overline{A + \overline{B} + C} + \overline{\overline{A} + B + \overline{C}}}$$

2.3.3. 2.3.3. ISKLJUČIVO ILI I ISKLJUČIVO NILI

Operacija isključivo ILI, skraćeno EX-ILI (*exclusive OR*) daje izlaz jednak stanju 1 ako se ulazne varijable razlikuju. Ako su ulazne varijable jednake, izlaz je u stanju 0. Operacija isključivo ILI naziva se još *antivalencija*[7].

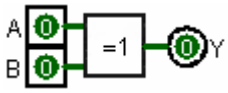
Operacija isključivo ILI

a) tablica stanja b) standardan simbol sklopa simbol sklopa prema IEC d) algebarski izraz

B	A	M_2
0	0	0
0	1	1
1	0	1
1	1	0

(a)

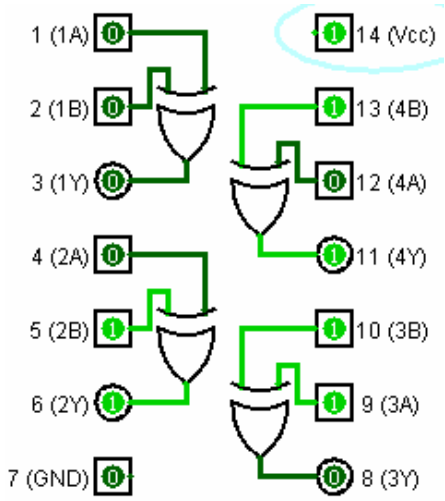
(b)



(c) $Y = A \oplus B$ (d)

Operacija isključivo ILI, odnosno sklop isključivo ILI, može se ostvariti logičkim zbrojem minterma ili logičkim umnoškom Maksterma. Danas se u praksi upotrebljavaju integrirane izvedbe (npr. 7486, 74C86, 74HC86).

Sklop 7486 četvorostruki EX-ILI



56. Primjer: Sklop za dozvolu i zabranu prolaza impulsa s 2 upravljačka ulaza.

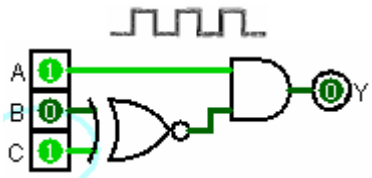
Impulsi mogu proći s ulaza na izlaz samo ako je jedan od dva upravljačka ulaza u stanju 1. U svim ostalim slučajevima izlaz sklopa ostaje u stanju 0.

Sklop I dopušta prolaz impulsa s jednoga ulaza na izlaz ako je drugi ulaz u stanju 1. Ako je drugi ulaz u stanju 0, tada je i izlaz u stanju 0 bez obzira na stanje signala na prvome ulazu. Sklop EX-ILI daje izlaz 1 ako su ulazi u različitim stanjima. Stoga na njegove ulaze treba dovesti upravljačke signale B i C. Izlaz sklopa EX-ILI veže se na jedan ulaz sklopa I s dva ulaza. Na drugi ulaz sklopa I dovode se

[7] *antivalencija* … suprotne vrijednosti

impulsi. Ako su upravljački signali jednaki, sklop EX-ILI daje na ulazu sklopa I stanje 0, što ne dopušta prolaz impulsa i daje na izlazu cijeloga sklopa stanje 0.

Sklop za dopuštenje i zabranu prolaza impulsa s dva upravljačka ulaza izveden uporabom sklopa EX-ILI



Operacija isključivo NILI, skraćeno EX-NILI (*exclusive NOR*), suprotna je operaciji isključivo ILI. To znači da će izlaz sklopa biti u stanju 1 ako su ulazne veličine međusobno jednake. Ako se ulazne veličine međusobno razlikuju, izlaz je u stanju 0.

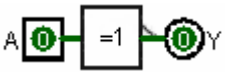
Operacija isključivo NILI

a) tablica stanja b) standardni simbol sklopa c) simbol sklopa IEC d) algebarski izraz

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

(a)

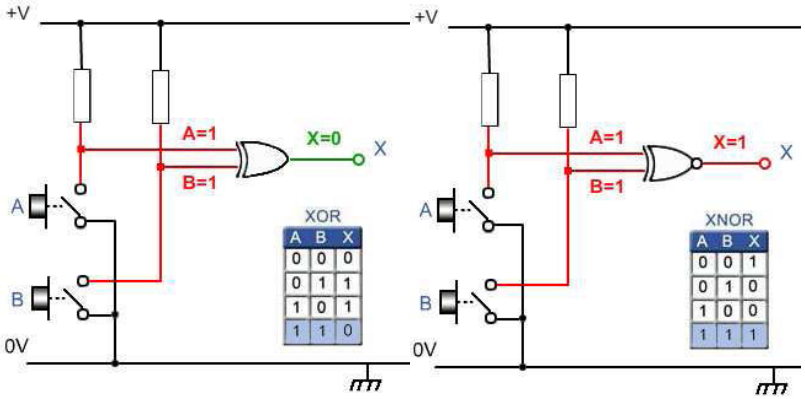
(b)



(c)

$Y = \overline{A \oplus B}$ (d)

Operacija isključivo NILI naziva se i *ekvivalencija*[8].



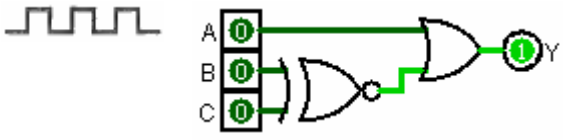
Logička operacija isključivo NILI, odnosno sklop isključivo NILI mogu se također ostvariti pomoću logičkih *umnožaka Maksterma* ili logičkoga *zbroja minterma*. Naravno, moguće je upotrijebiti i sklop isključivo ILI dodatkom invertora na izlazu. Osim toga postoje i integrirane izvedbe (npr. 74LS266, 74C266, 74HC266).

57. Primjer: Konstrukcija sklopa koji dopušta prolaz impulsa s dva upravljačka ulaza.

Impulsi mogu proći s ulaza na izlaz samo ako je jedan od dva upravljačka ulaza u stanju 1. U svim ostalim slučajevima izlaz sklopa treba biti u stanju 1. Sklop ILI dopušta prolaz impulsa s jednoga ulaza na izlaz kada je drugi ulaz u stanju 0. Ako je drugi ulaz u stanju 1, tada je i izlaz stalno u stanju 1 bez obzira na stanje signala.

Sklopovi koji mogu utvrditi jednakost ili različitost dvaju signala su EX-ILI i EX-NILI. Obzirom da sklop ILI za dopuštenje prolaza impulsa s jednoga ulaza na izlaz treba na drugome ulazu stanje 0, to za detektiranje upravljačkih signala treba uporabiti sklop EX-NILI, jer taj sklop daje na izlazu stanje 0 kada su upravljački signali različiti.

[8] *ekvivalencija* … jednako vrijedan, iste vrijednosti



2.3.4. 2.3.4. PREGLED KLJUČNIH POJMOVA

isključivo ILI (*exclusive* OR)

- logička operacija koja daje na izlazu stanje 1 ako su ulazne varijable različite

isključivo NILI (*exclusive* NOR)

- logička operacija koja daje na izlazu stanje 1 ako su ulazne varijable jednake

logički *umnožak* Maksterma (*the product-of-sums method*)

- postupak za realizaciju složenih logičkih operacija

logički *zbroj* minterma (*the sum-of-products method*)

- postupak za realizaciju složenih logičkih sklopova

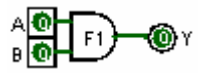
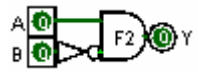
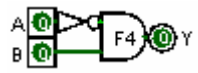
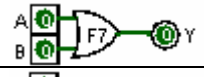
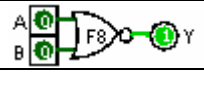
Maksterm

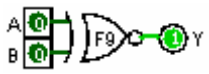
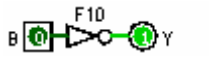
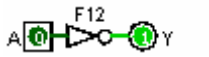
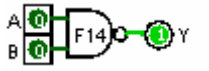
- logička operacija koja u izlaznome stupcu daje *samo jedanput* stanje 0

minterm

- logička operacija koja u izlaznome stupcu daje *samo jedanput* stanje 1

Tablica 2.1. Pregled mogućih funkcija s dvije varijable

Booleove funkcije ↓						Naziv funkcije	Algebarski izraz		Simbol
	A	0	1	0	1				
	B	0	0	1	1				
F_0	Y	0	0	0	0	konstanta	funkcija nema smisla, jer Y ne ovisi o A i B Nula ili clear, uvijek vraća nulu, bez obzira na ulazne vrijednosti A i B – poništavanje (*annihilation*)		
F_1	Y	0	0	0	1	AND	$A \cdot B$ Logičko AND = A*B. Vraća A AND B.		
F_2	Y	0	0	1	0	dozvola prolaza impulsa	$A \cdot \overline{B}$ Dozvola (*inhibition*) = AB' (A, NE B). Također ekvivalent je A>B ili B<A.		
F_3	Y	0	0	1	1	identitet A	A Kopira (*copy*) A. Vraća vrijednost broja A i zanemaruje vrijednost B. Ponekad se zove *prijenos* (*transfer*).		
F_4	Y	0	1	0	0	dozvola prolaza impulsa	$\overline{A} \cdot B$ Dozvola (*Inhibition*) = A'B (NE A, B). Također ekvivalent B>A ili A<B.		
F_5	Y	0	1	0	1	identitet B	B Kopira (*copy*) B. Vraća vrijednost B i zanemaruje vrijednost A. Ponekad se zove *prijenos* (*transfer*).		
F_6	Y	0	1	1	0	EX-ILI (antivalencija)	$A \oplus B$ Isključivo ILI (XOR): A⊕B. Također ekvivalentno A#B, A ili B, ali ne oba.		
F_7	Y	0	1	1	1	ILI	$A + B$ Logičko ILI = A + B. Vraća A ili B.		
F_8	Y	1	0	0	0	NILI	$\overline{A} \cdot \overline{B} = \overline{A + B}$ Logičko NOR (NE (A ili B)) = (A + B)'.		

Booleove funkcije ↓						Naziv funkcije	Algebarski izraz		Simbol
	A	0	1	0	1				
	B	0	0	1	1				
F_9	Y	1	0	0	1	EX-NILI (ekvivalencija)	$\overline{A \oplus B}$ Ekvivalencija: (A = B). Također poznata kao isključivo-NOR (NE isključivo-ILI).		
F_{10}	Y	1	0	1	0	negacija	$\overline{B}$ NE B. Vraća B' i ignorira A (B komplement).		
F_{11}	Y	1	0	1	1	implikacija	$\overline{A}+B$ Implikacija, B podrazumijeva A: A+B'. (Ako B onda A). Također ekvivalentno B>=A.		
F_{12}	Y	1	1	0	0	negacija	$\overline{A}$ NE A. Ignorira B i vraća A' (A komplement).		
F_{13}	Y	1	1	0	1	implikacija	$A+\overline{B}$ Implikacija, A implicira B: B + A' (ako A onda B). Također ekvivalentno A>=B.		
F_{14}	Y	1	1	1	0	NI	$\overline{A}+\overline{B} = \overline{A \cdot B}$ Logičko NAND (NE (A AND B)) = (A * B)'.		
F_{15}	Y	1	1	1	1	konstanta	funkcija nema smisla, jer Y ne ovisi o A i B Jedan ili postavka. Uvijek vraća 1 bez obzira na ulazne vrijednosti A i B jediničnost (*identity*).		

2.3.5. 2.3.5. PITANJA I ZADACI ZA PONAVLJANJE

1. Objasnite pojam logičke operacije minterm.

   minterm, broj jedinica na izlazu je minimalan

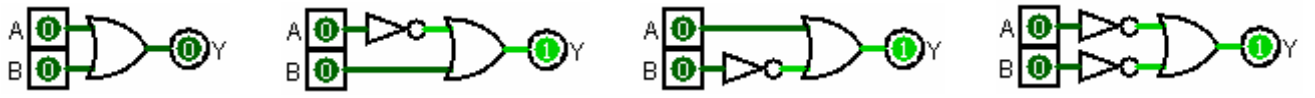
2. Kako je moguće realizirati sklop za operaciju minterm?

(a) (b) (c) (d)

3. Objasnite pojam operacije Maksterm.

   Maksterm, broj jedinica na izlazu je maksimalan

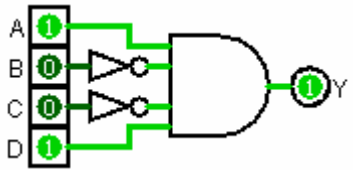
4. Kako je moguće realizirati sklop za operaciju Maksterm?



(a) (b) (c) (d)

5. Napišite tablicu stanja, algebarski izraz i logičku shemu za minterm m_9 s četiri ulazne varijable.

D	C	B	A	Y	minterm
0	0	0	0	0	m_0
0	0	0	1	0	m_1
0	0	1	0	0	m_2
0	0	1	1	0	m_3
0	1	0	0	0	m_4
0	1	0	1	0	m_5
0	1	1	0	0	m_6
0	1	1	1	0	m_7
1	0	0	0	0	m_8
1	0	0	1	1	m_9

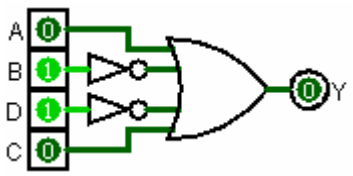


$Y = A \cdot \overline{B} \cdot \overline{C} \cdot D$

1	0	1	0	0	m_{10}
1	0	1	1	0	m_{11}
1	1	0	0	0	m_{12}
1	1	0	1	0	m_{13}
1	1	1	0	0	m_{14}
1	1	1	1	0	m_{15}

6. Napišite tablicu stanja, algebarski izraz i logičku shemu za Maksterm M_{10} s četiri ulazne varijable.

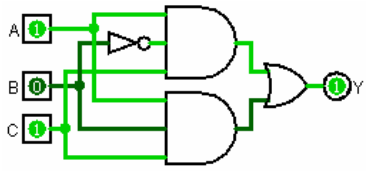
D	C	B	A	Y	Maksterm
0	0	0	0	1	M_0
0	0	0	1	1	M_1
0	0	1	0	1	M_2
0	0	1	1	1	M_3
0	1	0	0	1	M_4
0	1	0	1	1	M_5
0	1	1	0	1	M_6
0	1	1	1	1	M_7
1	0	0	0	1	M_8
1	0	0	1	1	M_9
1	0	1	0	0	M_{10}
1	0	1	1	1	M_{11}
1	1	0	0	1	M_{12}
1	1	0	1	1	M_{13}
1	1	1	0	1	M_{14}
1	1	1	1	1	M_{15}



$Y = \overline{A} + B + \overline{C} + D$

7. Napišite tablicu stanja i algebarski izraz (logički zbroj minterma) te nacrtajte logičku shemu za operaciju koja daje izlaz Y = 1 ako su ulazne varijable A = 1, B = 1, C = 1 i A = 1, B = 0, C = 1. Provedite postupak minimizacije i nacrtajte logičku shemu sklopa s minimalnim brojem elemenata.

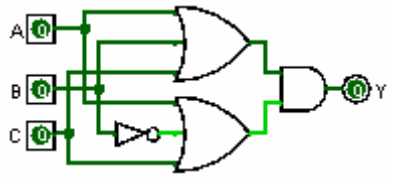
C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$Y = A\overline{B}C + ABC = AC$

8. Napišite tablicu stanja i algebarski izraz (logički umnožak Maksterma) te nacrtajte logičku shemu za operaciju koja daje izlaz Y = 0 ako su ulazne varijable A = 0, B = 0, C = 0 i A = 0, B = 1, C = 0. Provedite postupak minimizacije i nacrtajte logičku shemu sklopa s minimalnim brojem elemenata.

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

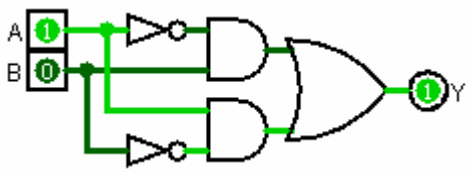


$Y = (A+B+C)(A+\overline{B}+C) = A+C$

9. Objasnite pojam operacije isključivo ILI, napišite algebarski izraz, tablicu stanja i simbol sklopa.

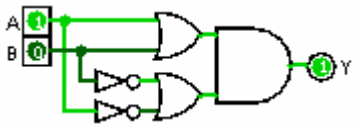
10. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo ILI ostvarenoga pomoću logičkoga zbroja minterma.

$Y = \overline{A} \cdot B + A \cdot \overline{B}$



11. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo ILI ostvarenoga pomoću logičkoga umnoška Maksterma.

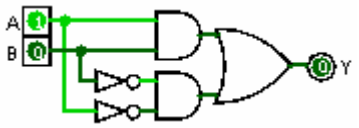
$Y = (\overline{A} + \overline{B}) \cdot (A + B)$



12. Objasnite pojam logičke operacije isključivo NILI, napišite tablicu stanja i simbol sklopa.

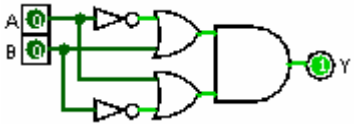
13. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo NILI ostvarenoga pomoću logičkoga zbroja minterma.

$Y = (\overline{A} \cdot \overline{B}) + (A \cdot B)$



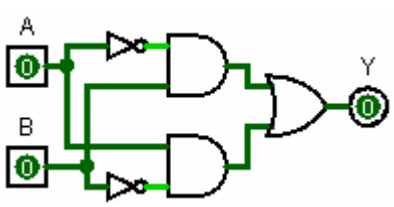
14. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo NILI ostvarenoga pomoću logičkoga umnoška Maksterma.

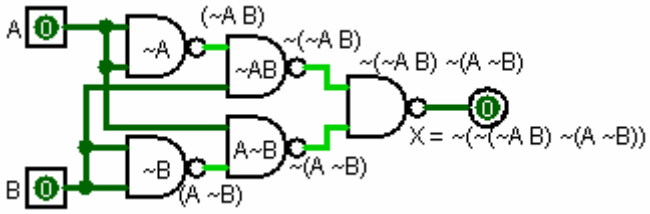
$Y = (\overline{A} + B) \cdot (A + \overline{B})$



15. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo ILI ostvarenoga korištenjem samo sklopova NI.

X = (~AB)+(A~B) = ~(~(~AB)~(A~B))



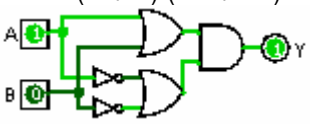


16. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo ILI ostvarenoga korištenjem samo sklopova NILI.

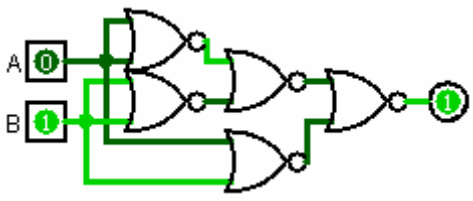
$Y = (A + B) \cdot$

$(\overline{A} + \overline{B}) = \overline{\overline{A + B} + \overline{A} + \overline{B}}$

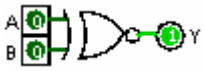
$Y = (A + B)\ (\sim A + \sim B)$



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

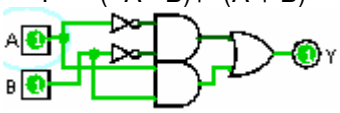


17. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo NILI ostvarenoga korištenjem samo sklopova NI.

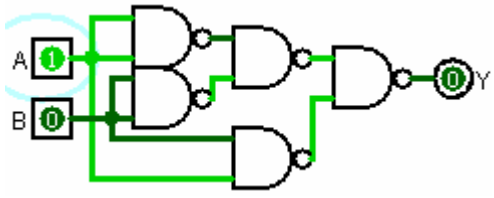


$$Y = \overline{(\overline{A \cdot \overline{B}}) + (A \cdot B)} = \overline{\overline{A \cdot \overline{B}}} \cdot \overline{\overline{B \cdot A}}$$

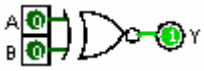
Y = ~(~A ~B)+~(A + B)



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

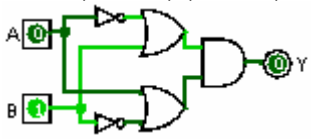


18. Napišite algebarski izraz i nacrtajte logičku shemu sklopa isključivo NILI ostvarenoga korištenjem samo sklopova NILI.

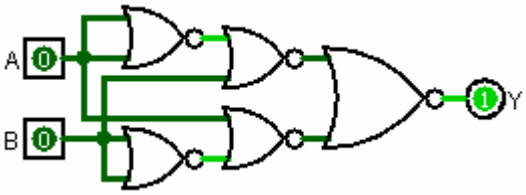


$$Y = \overline{(\overline{A} + B) \cdot (A + \overline{B})} = \overline{\overline{\overline{A} + BA}} \cdot \overline{\overline{B}}$$

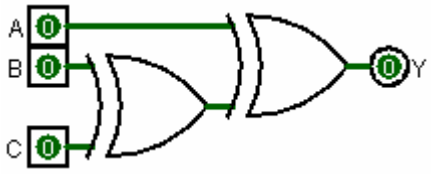
Y = (~A + B) (A + ~B)



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

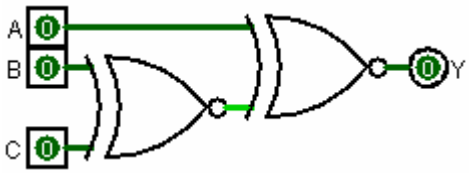


19. Za koje ulazne kombinacije sklopa je izlaz Y u stanju 1?



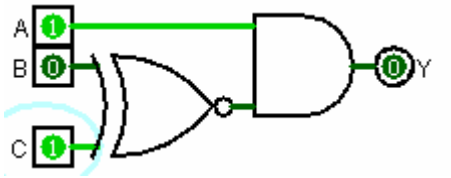
Za kombinacije s neparnim brojem jedinica.

20. Za koje ulazne kombinacije sklopa je izlaz Y u stanju 0?

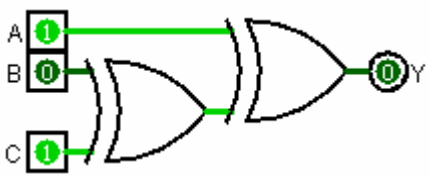


Za kombinacije s parnim brojem jedinica.

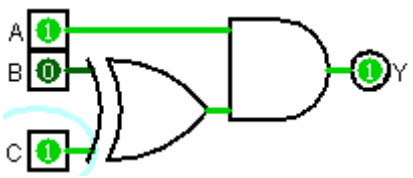
21. Nacrtajte logičku shemu sklopa koji će dopustiti prolaz impulsa s ulaza A ako su upravljački signali B i C jednaki. Ako su upravljački signali B i C različiti, izlaz Y sklopa treba biti u stanju 0.



22. Nacrtajte logičku shemu sklopa koji će dopustiti prolaz impulsa s ulaza A ako su upravljački signali B i C jednaki. Ako su upravljački signali B i C različiti, izlaz Y sklopa treba biti u stanju 0.



23. Nacrtajte logičku shemu sklopa koji će dopustiti prolaz impulsa s ulaza A ako su upravljački signali B i C različiti.



58. Sljedeća znamenka je 10.exe (gotov program) - povratak

```
//#include <conio.h> //_getch();
```

```
#include<iostream>
using namespace std;
int main(){
int broj,duzina,baza;
char polje[33];//umjesto običnoga polja,
koristiti vector
//Problem veličine polja[] odražava se tek
//pri izlasku iz programa (vidi napomenu na
dnu!)
cout<<"Za izlaz, upisi 1 ili 0\n";
do{
broj=0;
cout<<"Upisi cio dekadski broj: ";
cin>>broj;
if(broj==1||broj==0)return(0);
//duzina=strlen(char* broj);
cout<<"Upisi bazu sustava: ";
cin>>baza;
if(baza==1||baza==0)return(0);
//cout<<" "<<broj<<endl;
if(broj==baza)cout<<"Uz broj==baza, rezultat
je: ";
cout<<itoa(broj,polje,baza)<<endl;
}while(getchar()!=EOF);
//putchar(c);
return(0);
}
```

```
C:\windows\system32\cmd.exe
Za izlaz, upisi 1 ili 0
Upisi cio dekadski broj: 5
Upisi bazu sustava: 5
Uz broj==baza, rezultat je: 10
Upisi cio dekadski broj: 4
Upisi bazu sustava: 4
Uz broj==baza, rezultat je: 10
Upisi cio dekadski broj: 3
Upisi bazu sustava: 3
Uz broj==baza, rezultat je: 10
Upisi cio dekadski broj: 2
Upisi bazu sustava: 2
Uz broj==baza, rezultat je: 10
Upisi cio dekadski broj: 1
Press any key to continue . . .
```

3. LITERATURA

[1.] Charles W. Kann III, *Digital Circuit Projects*, 277 E. Lincoln Ave., Gettysburg, Pa,

[2.] M. Morris Mano, Michael D. Ciletti, *Digital Design With an Introduction to the Verilog HDL*, ISBN-13: 978-0-13-277420-8, Pearson, 2013.

[3.] Stanko Paunović: *Digitalna elektronika 1, 1. svezak, brojevni sustavi i kodovi, logički sklopovi, skupine integriranih digitalnih sklopova, multivibratori*, 2. izdanje, Školska knjiga Zagreb, 1999.

[4.] George Self, *Lab Manual For Exploring Digital Logic with Logisim*, Cochise College, 901 N. Colombo Ave, Sierra Vista, AZ 85650, First Edition: Summer, 2014.

[5.] George Self, *Exploring Digital Logic with Logisim*, Cochise College, 901 N. Colombo Ave, Sierra Vista, AZ 85650, First Edition: Summer, 2013.

[6.] Aleksandar Szabo, *Impulsna i digitalna elektronika I i II.*, Školski centar "Ruđer Bošković", Zagreb, 1973.

[7.] Daniel J. Tylaysky, *Digital Design for the Laboratory: Hardware & Simulation (using LogicWorksTM 5) version 1.*, ISBN 0-9662944-0-8, Center Point Publishing, 2009.