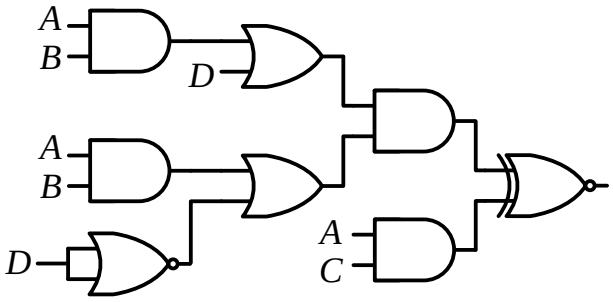
# Osnovi digitalne elektronike (13E042OD)

- VEŽBE -

## Kombinacione mreže i prekidačka algebra

**1** Koristeći NMOS i PMOS tranzistore, isprojektovati jednostepeno statičko CMOS kolo koje realizuje Bulovu funkciju kao i kolo sa slike, a pri tome ima minimalan broj tranzistora. Minimizaciju raditi algebarskom minimizacijom.




**Rešenje:**

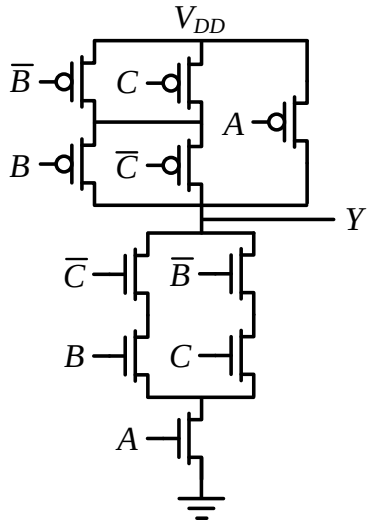
Najpre obratiti pažnju na poslednje logičko kolo u nizu. To je XNOR logičko kolo (ekskluzivno NILI) koje realizuje je negaciju XOR logičke funkcije:

$\overline{X \oplus Y} = \overline{X\overline{Y} + \overline{X}Y} = \left(\overline{X} + Y\right)\left(X + \overline{Y}\right) = XY + \overline{X}\,\overline{Y}$.

Logička funkcija koju realizuje kolo sa slike se može minimizovati primenom aksioma i teorema Bulove algebre:

$$Y = \overline{\left((AB + D)\left(AB + \overline{D}\right)\right) \oplus (AC)} = \overline{\left(AB + ABD + AB\overline{D}\right) \oplus (AC)} = \overline{\left(AB\left(1 + D + \overline{D}\right)\right) \oplus (AC)} =$$
$$= \overline{(AB) \oplus (AC)} = \overline{AB\,\overline{AC} + \overline{AB}AC} = \overline{AB\left(\overline{A} + \overline{C}\right) + \left(\overline{A} + \overline{B}\right)AC} = \overline{AB\overline{A} + AB\overline{C} + \overline{A}AC + \overline{B}AC} =$$
$$= \overline{AB\overline{C} + \overline{B}AC} = \overline{A\left(B\overline{C} + \overline{B}C\right)}$$

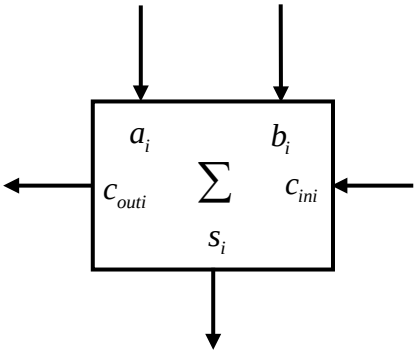
Jednostepeno CMOS kolo koje realizuje ovu funkciju je dato na slici 1.1.




*Slika 1.1 – Jednostepeno statičko CMOS logičko kolo koje realizuje funkciju Y*

**2** a) Koristeći što manji broj dvoulaznih logičkih kola izvršiti sintezu jedne ćelije potpunog sabirača. Na ulaz ćelije sabirača dovode se biti $a_i$ i $b_i$ razreda $i$, koje treba sabrati, kao i bit prenosa $c_{in,i}$ iz prethodnog razreda. Izlaz ćelije sabirača daje rezultat sabiranja u $i$-tom razredu, $s_i$, kao i prenos $c_{out,i}$ u naredni razred.

b) Koristeći ćelije potpunog sabirača iz prethodne tačke i potrebna logička kola, izvršiti sintezu kombinacione mreže za oduzimanje označenih brojeva predstavljenih u komplementu dvojke sa četiri bita. Potrebno je generisati i signal $g$, koji ukazuje na to da je došlo do greške prilikom oduzimanja označenih brojeva (*overflow*), sa ulaza mreže.




**Rešenje:**

Blok šema jedne ćelije potpunog sabirača je prikazana na slici uz tekst zadatka. Na ulaz bloka se dovode signali $a_i$ , $b_i$, $c_{ini}$, a na izlazu se dobijaju signali $s_i$, $c_{outi}$. Opis preslikavanja vektora ($a_i$ , $b_i$, $c_{ini}$) → ($s_i$, $c_{outi}$) dat je u tabeli 2.1.

*Tabela* 2.1 – *Preslikavanje ulaznog vektora u izlazni vektor*

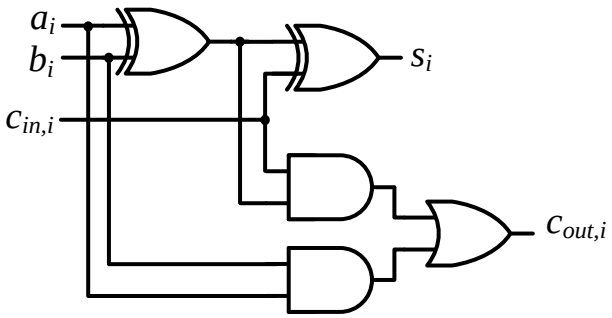
| $a_i$ | $b_i$ | $c_{ini}$ | $s_i$ | $c_{outi}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Algebarskom minimizacijom dolazi se do sledećih izraza za izlaze bloka potpunog sabirača:

$$s_i = \overline{a_i}\,\overline{b_i}c_{in,i} + \overline{a_i}b_i\overline{c_{in,i}} + a_i\overline{b_i}\,\overline{c_{in,i}} + a_ib_ic_{in,i} = \overline{a_i}\left(\overline{b_i}c_{in,i} + b_i\overline{c_{in,i}}\right) + a_i\left(\overline{b_i\,\overline{c_{in,i}}} + b_ic_{in,i}\right) =$$
$$= \overline{a_i}\left(b_i \oplus c_{in,i}\right) + a_i\overline{\left(b_i \oplus c_{in,i}\right)} = a_i \oplus b_i \oplus c_{in,i}$$

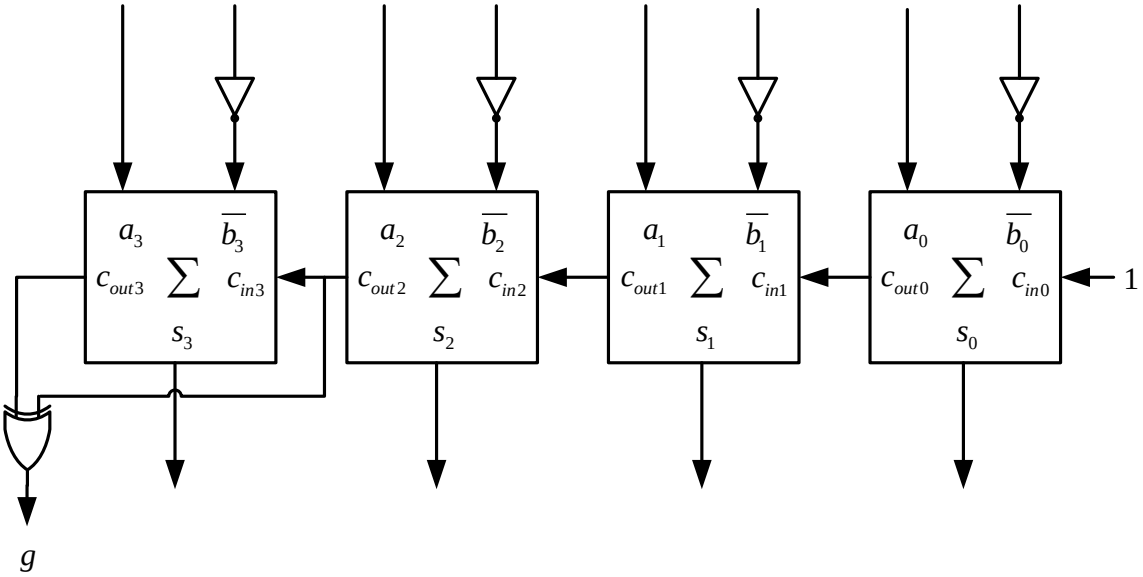
$$c_{out,i} = \overline{a_i}b_ic_{in,i} + a_i\overline{b_i}c_{in,i} + a_ib_i\overline{c_{in,i}} + a_ib_ic_{in,i} = c_{in,i}\left(\overline{a_i}b_i + a_i\overline{b_i}\right) + a_ib_i\left(\overline{c_{in,i}} + c_{in,i}\right) =$$
$$= \left(a_i \oplus b_i\right)c_{in,i} + a_ib_i$$

Primetiti da u izrazima postoji zajedničko XOR kolo: $a_i \oplus b_i$, pa se ta pogodnost može iskoristiti da se navedene funkcije realizuju koristeći još manji broj kola. Realizacija potpunog sabirača je prikazana na slici 1.1.




*Slika 1.1 – Realizacija potpunog sabirača*

Oduzimanje brojeva u komplementu dvojke se može prikazati kao $a - b = a + (-b) = a + \overline{b} + 1$ , gde je $\overline{b}$ , bit po bit komplementirana vrednost binarnog broja *b*. Prema tome kombinaciona mreža za oduzimanje brojeva u komplementu dvojke sa četiri bita se može projektovati kao kaskadna veza četiri ćelije potpunog sabirača, gde se na svaki ulaz *i*-te ćelije dovodi $a_i$ i $b_i$, pri čemu je $c_{in,0}$ = 1, prenos iz nepostojećeg razreda, kako bi se dobio komplement broja *b*. Prekoračenje se može detektovati tako što se detektuje da su dva poslednja bita prenosa različita, za šta je pogodna XOR logička funkcija. Realizacija ove kombinacione mreže je prikazana na slici 1.2.




*Slika* 1.2 – *Četvorobitni oduzimač*

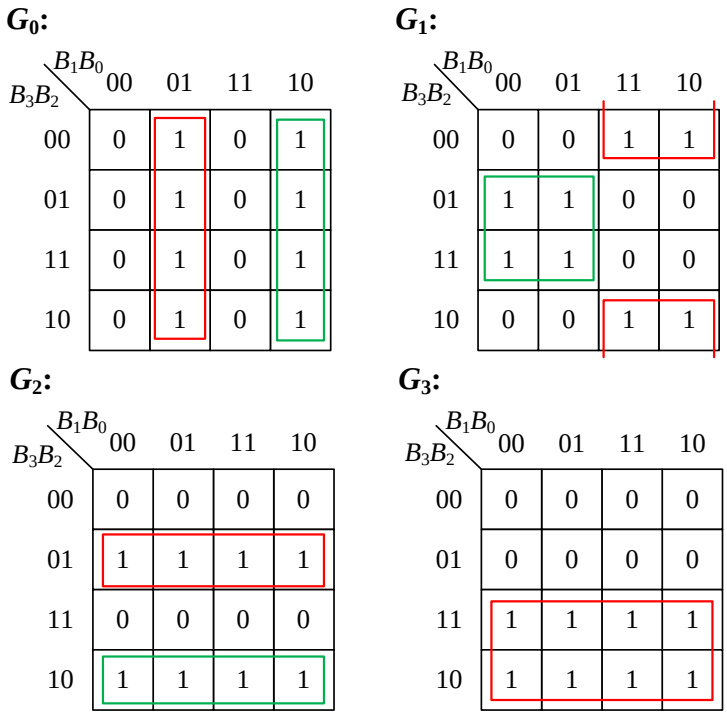
**3** Projektovati kombinacionu mrežu koja konvertuje 4-bitni neoznačeni binarni broj u komplementu dvojke u 4-bitni binarni broj u Grejovom binarnom kodu. Težiti da broj upotrebljenih logičkih kola bude minimalan.

**Rešenje:**

Najpre određujemo kombinacionu tabelu (funkcionalnu tabelu, tablicu istinitosti) koja definiše vrednosti izlaznih signala za sve kombinacije ulaznih logičkih nivoa. Postupak dobijanja Grejovog koda se može naći na sledećem linku, zadatak 1.

*Tabela 3.1 – Kombinaciona tabela konverzije iz binarnog u Grejov binarni kod*

| Binarni kod – kompl. dvojke ($B_3B_2B_1B_0$) | Grejov binarni kod ($G_3G_2G_1G_0$) |
|---|---|
| 0000 | 0000 |
| 0001 | 0001 |
| 0010 | 0011 |
| 0011 | 0010 |
| 0100 | 0110 |
| 0101 | 0111 |
| 0110 | 0101 |
| 0111 | 0100 |
| 1000 | 1100 |
| 1001 | 1101 |
| 1010 | 1111 |
| 1011 | 1110 |
| 1100 | 1010 |
| 1101 | 1011 |
| 1110 | 1001 |
| 1111 | 1000 |




*Slika* 3.1 – *Karnoove mape*

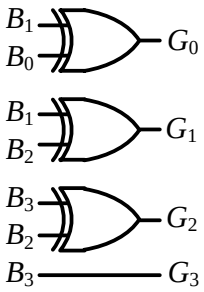
Funkcije izlaza $G_3G_2G_1G_0$ su logičke funkcije ulaznih bita $B_3B_2B_1B_0$ i određujemo ih pojedinačno iz Karnoove mape za svaki bit $G_3$-$G_0$. Na slici 3.1 su prikazane Karnoove mape za svaki od izlaza $G_3$-$G_0$, a logičke funkcije koje odgovaraju ovim mapama su:

$$\boldsymbol{G_0 = \overline{B_1} \cdot B_0 + B_1 \cdot \overline{B_0} = B_1 \oplus B_0}$$

$$\boldsymbol{G_1 = \overline{B_1} \cdot B_2 + B_1 \cdot \overline{B_2} = B_1 \oplus B_2}$$

$$\boldsymbol{G_2 = \overline{B_3} \cdot B_2 + B_3 \cdot \overline{B_2} = B_3 \oplus B_2}$$

$$\boldsymbol{G_3 = B_3}$$




*Slika 3.2 – Realizacija konverzije iz binarnog u Grejov binarni kod*

**4** Projektovati kombinacionu mrežu kojom se realizuje funkcija $Y = \overline{CB}A + C\overline{B} + CB\overline{A}$:

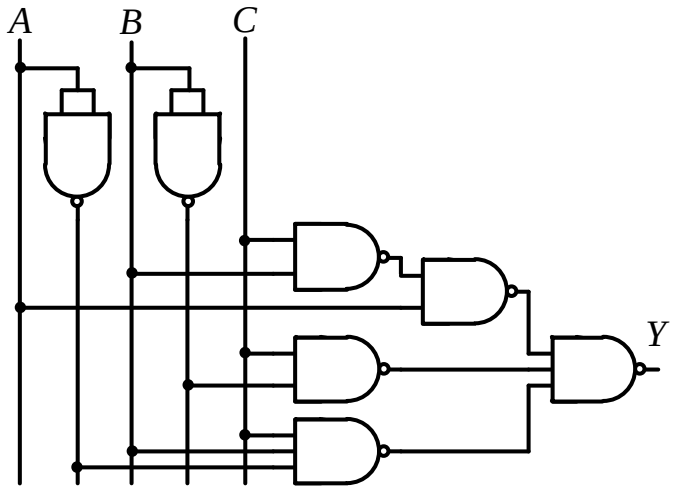
a) Ako su na raspolaganju samo NI logička kola sa proizvoljnim brojem ulaza
b) Ako su na raspolaganju samo dvoulazna NI logička kola
c) Ako su na raspolaganju samo dvoulazna NILI logička kola
d) Ako je na raspolaganju samo multiplekser 8 u 1 (napraviti *look up* tabelu).
e) Ako je na raspolaganju multiplekser 4 u 1 i potreban broj osnovnih logičkih kola
f) Ako su na raspolaganju samo multiplekseri 2 u 1, najpre napraviti multiplekser 8 u 1, a zatim ponoviti tačku d.

**Rešenje:**

a) Funkcija *Y* je data u disjunktivnoj formi, pa se može realizovati korišćenjem NI logičkih kola. Dvostrukim komplementiranjem i primenom Demorganove teoreme dobija se izraz:

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

Funkcija u ovom obliku se može realizovati korišćenjem dva troulazna NI kola, tri dvoulazna NI kola i dva invertora. Kako su nam na raspolaganju samo NI logička kola, invertor se može realizovati na dva načina – spajanjem ulaza i korišćenjem jednog ulaza, pri čemu je drugi ulaz preko *pull up* otpornika povezan na napajanje (logičku jedinicu). Realizacija zadate funkcije je prikazana na slici 4.1.




*Slika* 4.1 – *Realizacija funkcije Y pomoću NI logičkih kola*

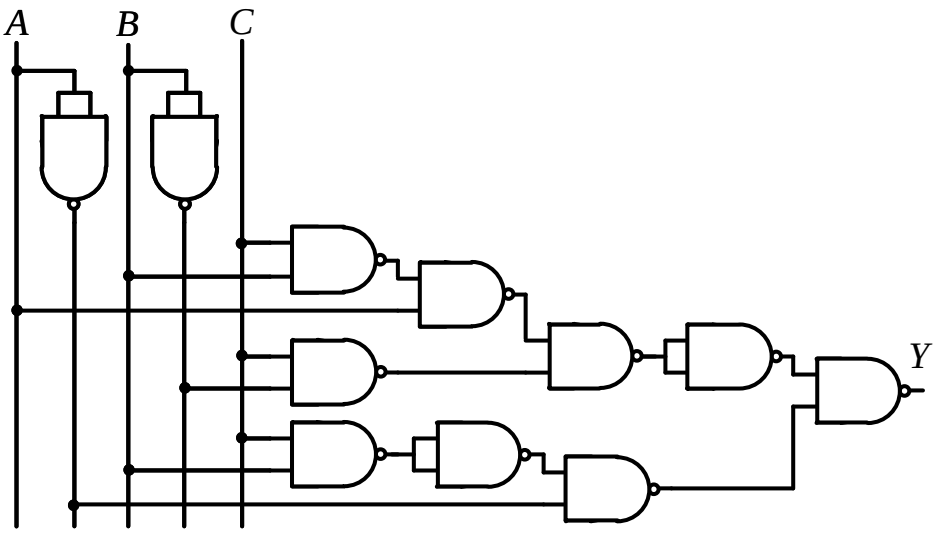
b) Funkciju realizovanu u tački a je potrebno dodatno transformisati da bi se ona realizovala korišćenjem samo dvoulaznih NI logičkih kola. S obzirom na to da je u tački a, funkcija realizovana korišćenjem dvoulaznih i troulaznih logičkih kola, korisno je videti na opštem primeru kako se logička funkcija koju realizuje troulazno NI kolo, realizuje uz pomoć dvoulaznih NI kola. Neka je $F = \overline{ABC}$. Funkcija *F* se realizuje tako što se grupišu dva člana iz logičkog proizvoda i njihov proizvod se dvostruko komplementira:

$$F = \overline{ABC} = \overline{(AB)C} = \overline{\overline{A \cdot B} \cdot C}.$$

Na osnovu toga je:

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

Realizacija zadate funkcije je prikazana na slici 4.2.




*Slika* 4.2 – *Realizacija funkcije Y pomoću dvoulaznih NI logičkih kola*

c) Funkcija *Y* je data u disjunktivnoj formi, pa se u ovom obliku ne može realizovati uz pomoć NILI logičkih kola. Da bi se to omogućilo, funkcija se mora predstaviti u konjuktivnoj formi. Ovo je najlakše uraditi pomoću Karnoove mape.

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

*Slika* 4.3 – *Karnoova mapa za logičku funkciju Y*

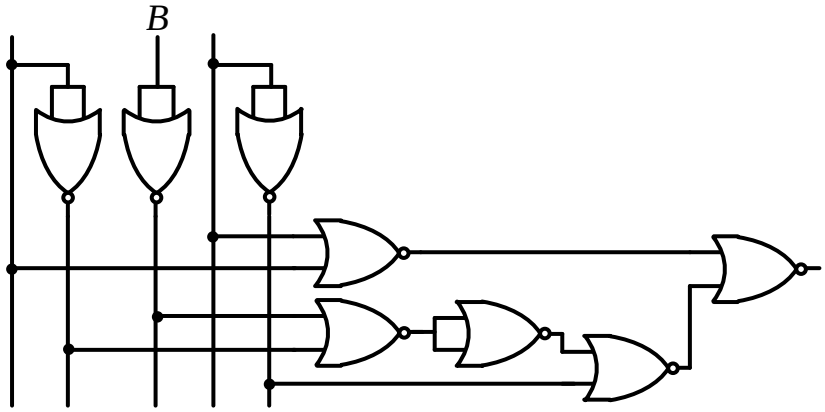
Uočavanjem najvećih mogućih kontura koje obuhvataju logičke nule, dolazi se do sledeće funkcije u konjuktivnoj formi:

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

Dvostrukim komplementiranjem ove funkcije, primenom Demorganovih zakona i grupisanjem odgovarajućih promenljivih dobija se funkcija koja se može realizovati samo uz pomoć dvoulaznih NILI logičkih kola:

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

Realizacija ove funkcije prikazana je na slici 4.4.

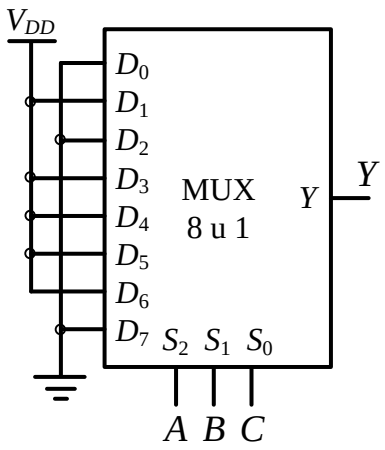



*Slika* 4.4 – *Realizacija funkcije Y pomoću dvoulaznih NILI logičkih kola*

d) Realizacija uz pomoć multipleksera 8 u 1 predstavlja direktnu realizaciju tablice istinitosti gde su ulazne promenljive zapravo selekcioni signali multipleksera, a ulazi multipleksera vrednosti logičke funkcije za različite kombinacije vrednosti ulaza. Ovakva realizacija logičke funkcije se naziva *look up* tabela i vrlo je česta u programabilnim kolima. Tablica istinitosti je data u tabeli 4.1, a realizacija funkcije uz pomoć multipleksera 8 u 1 na slici 4.5.

*Tabela* 4.1- *Tablica istinitosti funkcije Y*

| *A* | *B* | *C* | *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 | 0 |




*Slika* 4.5 – *Realizacija funkcije Y uz pomoć multipleksera 8 u 1*

e) S obzirom na to da se *look up* tabela 3 promenljive može napraviti multiplekserom sa najmanje $2^3$ ulaza, za realizaciju logičke funkcije *Y* pomoću četvoroulaznog multipleksera, potrebno je odabrati dve ulazne promenljive za selekcione signale, a treću preko određenih logičkih kola dovesti na neke od ulaza multipleksera. Najlakše je na selekcione ulaze priključiti promenljive koje su u logičkoh funkciji zastupljene sa najviše logičkih kombinacija. Za funkciju $Y = \overline{CB}A + C\overline{B} + CB\overline{A}$ su to signali *B* i *C*, pa ćemo na selekcioni ulaz $S_0$ povezati signal *C*, a na selekcioni ulaz $S_1$ povezati signal *B*. Kada je selektovan jedan od ulaza $D_i$, logička funkcija na izlazu multipleksera je određena izrazom: $D_i = Y(A, S_0 = C, S_1 = B)$, pa je za datu funkciju:

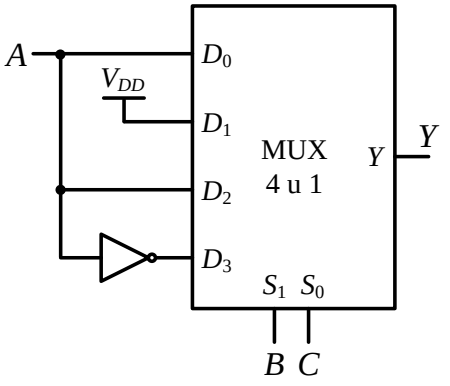
$$D_0 = Y(A, S_0 = C = 0, S_1 = B = 0) = \overline{(0 \cdot 0)} \cdot A + 0 \cdot 1 + 0 \cdot 0 \cdot \overline{A} = A,$$

$$D_1 = Y(A, S_0 = C = 1, S_1 = B = 0) =$$
$$= \overline{(1 \cdot 0)} \cdot A + 1 \cdot 1 + 1 \cdot 0 \cdot \overline{A} = 1,$$

$$D_2 = Y(A, S_0 = C = 0, S_1 = B = 1) =$$
$$= \overline{(0 \cdot 1)} \cdot A + 0 \cdot 0 + 0 \cdot 1 \cdot \overline{A} = A,$$

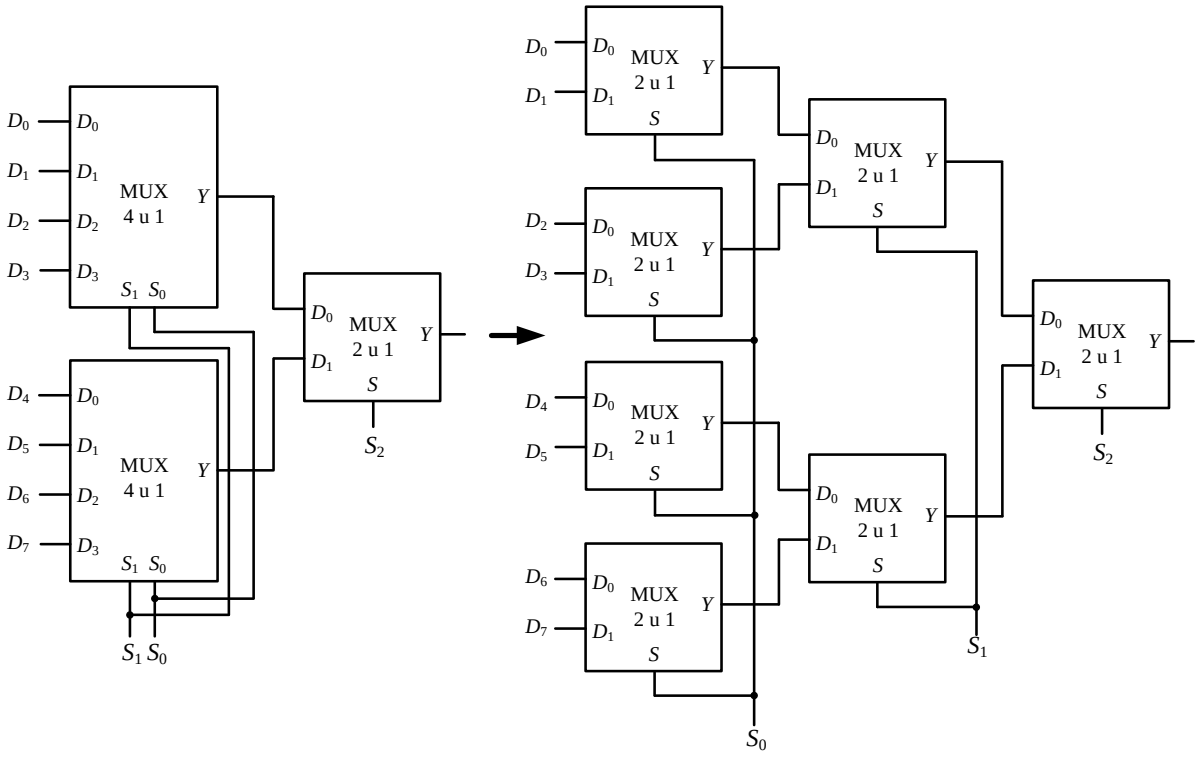
$$D_3 = Y(A, S_0 = C = 1, S_1 = B = 1) =$$
$$= \overline{(1 \cdot 1)} \cdot A + 1 \cdot 0 + 1 \cdot 1 \cdot \overline{A} = \overline{A}.$$

Realizacija funkcije *Y* uz pomoć multipleksera 4 u 1 i jednog invertora je data na slici 4.6.




*Slika* 4.6 – *Realizacija funkcije Y uz pomoć multipleksera 4 u 1 i jednog invertora*

f) Multiplekser 8 u 1 se može napraviti uz pomoć 2 multipleksera 4 u 1 i jednog multipleksera 2 u 1, a dalje se multiplekser 4 u 1 može napraviti uz pomoć 3 multipleksera 2 u 1. Realizacija multipleksera 8 u 1 uz pomoć multipleksera 2 u 1 je data na slici 4.7, a logička funkcija se realizuje kao look up tabela na isti način kao u tački d ovog zadatka.

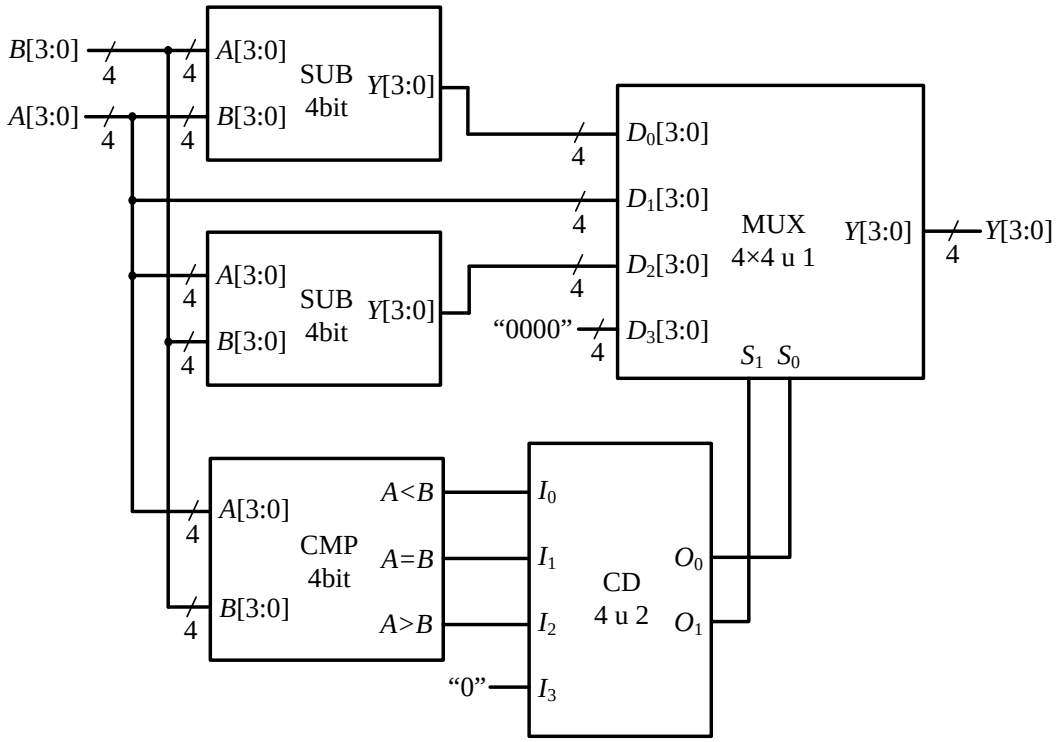



*Slika* 4.7 – *Realizacija multipleksera 8 u 1 pomoću multipleksera 2 u 1*

**5** Koristeći multipleksere, oduzimače iz zadatka 2, potpuni binarni koder i *magnitude* komparator koji ima tri izlaza: $A=B$, $A>B$ i $A<B$, projektovati kombinacionu mrežu koja na ulazima ima dva neoznačena 4-bitna broja $A[3:0]$ i $B[3:0]$, a na izlaz propušta broj $A$ ukoliko je $A=B$, broj $A–B$ ukoliko je $A>B$ i broj $B–A$ ukoliko je $B>A$.

**Rešenje:**

Za poređenje brojeva $A$ i $B$ iskoristićemo 4-bitni *magnitude* komparator. Ovaj komparator daje tri signala na izlazu koji određuju koji rezultat treba propustiti na izlaz. S obzirom na to da se za selekciju koji rezultat treba propustiti najčešće koristi multiplekser, najlakše je navedena tri signala kodovati u dvobitni podatak koristeći potpuni binarni koder, a taj podatak dovesti na selekcione ulaze multipleksera 4 puta 4 u 1. Nekorišćeni ulazi se moraju povezati na stabilan logički nivo.



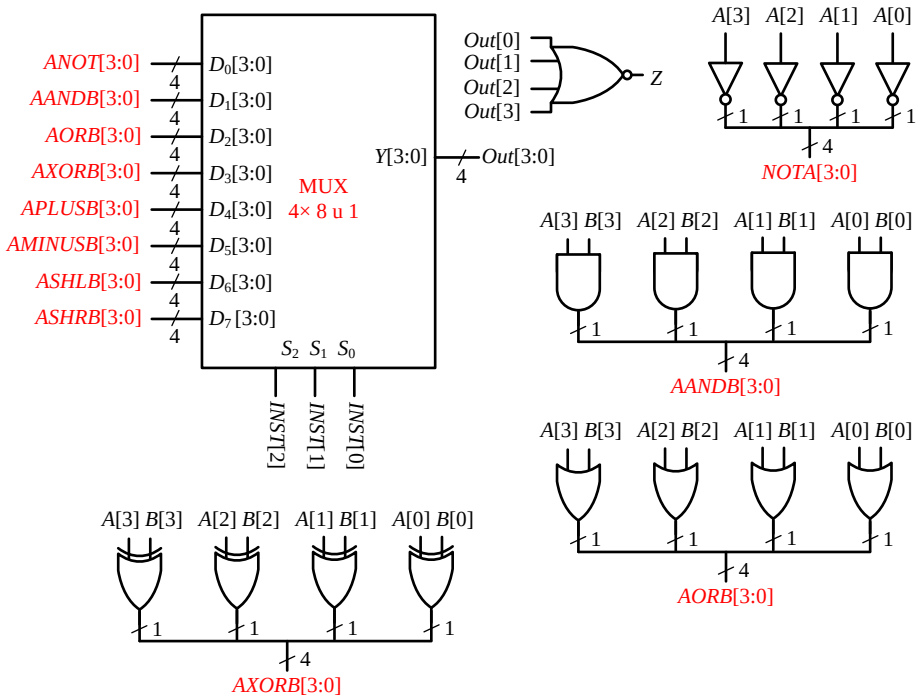

*Slika 5.1 – Kombinaciona mreža uz zadatak 5*

**6** Potrebno je projektovati aritmetičko-logičku jedinicu (*Arithmetic Logic Unit* – ALU) za 4-bitni procesor koji se koristi u dečijim igračkama. Ulazi u aritmetičko-logičku jedinicu su dva 4-bitna broja *A*[3:0] i *B*[3:0] i trobitni podatak *INST*[2:0] koji označava koju instrukciju (aritmetičku ili logičku operaciju) je potrebno izvršiti. Procesor ima 8 instrukcija navedenih u tabeli. Ako processor treba da izvrši instrukciju NOT, na izlazu ALU je potrebno generisati invertovan broj *A*, tj. sve njegove pojedinačne bite. Ako je u pitanju instrukcija AND, na izlazu ALU je potrebno generisati 4-bitni podatak koji je rezultat bitske I operacije $A \cdot B$. Slično, ako je u pitanju instrukcija OR ili XOR, na izlazu ALU je potrebno generisati 4-bitni podatak koji je rezultat bitske OR ili XOR operacije $A+B$ ili $A \oplus B$. Ukoliko procesor izvršava instrukciju ADD ili SUB, na izlazu je potrebno generisati rezultat sabiranja odnosno oduzimanja brojeva *A* i *B*. Instrukcija SHL (*SHift Left*) podrazumeva pomeranje svih bita broja *A* ulevo za onoliko mesta kolika je vrednost broja *B*. Instrukcija SHR (*SHift Right*) podrazumeva pomeranje svih bita broja *A* udesno za onoliko mesta kolika je vrednost broja *B*. Smatrati da je u ova dva slučaja broj *B* pozitivan i ne veći od 3. Pored ovoga, ALU generiše signal Z koji ima vrednost 1, ako je rezultat na izlazu ALU jednak nuli i signal C koji ima vrednost 1, ako je izlazni prenos prilikom sabiranja ili oduzimanja jednak 1. Na raspolaganju je samo jedan 4-bitni sabirač i proizvoljan broj osnovnih logičkih kola i ostalih kombinacionih integrisanih kola.

| Instrukcija | NOT | AND | OR | XOR | ADD | SUB | SHL | SHR |
|---|---|---|---|---|---|---|---|---|
| *INST*[2:0] | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

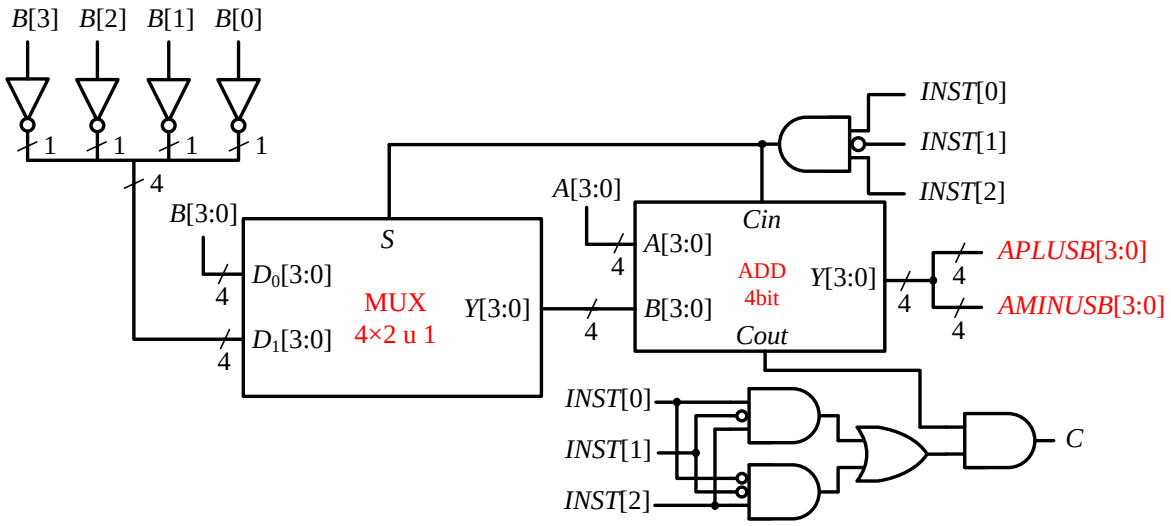
**Rešenje:**

Zadatak je najlakše rešiti tako što se izgenerišu rezultati za sve instrukcije, a onda se samo višestrukim multiplekserom 8 u 1 odabere rezultat na osnovu zadate instrukcije. Logičke operacije i odabir rezultata su prikazani na slici 6.1. Signal *Z* se generiše ako je rezultat jednak nuli, pa je za njegovo generisanje pogodno NILI kolo.




*Slika* 6.1 – *Odabir rezultata, logičke operacije i generisanje signala Z*

S obzirom na to da nam je na raspolaganju samo jedan 4-bitni sabirač. Potrebno je njegovim korišćenjem izvršiti operacije i sabiranja i oduzimanja. U zadatku 2 smo videli da se oduzimanje može uraditi tako što se drugi sabirak invertuje i na ulazni prenos dovede logička jedinica. Sabirač ima dva sabirka *A* i *B*. Na ulaz *A* sabirača dovodimo broj *A*, dok na ulaz *B* sabirača dovodimo broj *B* u slučaju da se izvršava operacija sabiranja, a broj $\overline{B}$ u slučaju da se izvršava operacija oduzimanja. Takođe, na ulazni prenos je potrebno dovesti logičku nulu kod sabiranja, a logičku jedninicu kod oduzimanja. Za odabir koji broj dovodimo na ulaz *B* sabirača koristimo multiplekser 4×2 u 1. Na selekcioni signal multipleksera je potrebno dovesti logičku jedinicu, ako se izvršava operacija oduzimanja, pa je nju potrebno prepoznati. Kod operacije oduzimanja je *INST*[2:0] = 101, pa se troulaznim I kolom kod koga je invertovan bit *INST*[1] generiše logička jedinica kada je *INST*[2:0] = 101. Na slici 6.2 prikazane su operacije sabiranja i oduzimanja. Izlazni prenos generiše signal *C*, ali samo kada je potrebno izvršiti operacije sabiranja ili oduzimanja, što je regulisano prepoznavanjem instrukcije sabiranja (*INST*[2:0] = 100) ili instrukcije oduzimanja (*INST*[2:0] = 101).




*Slika* 6.2 – *Operacije sabiranja i oduzimanja i generisanje signala C*

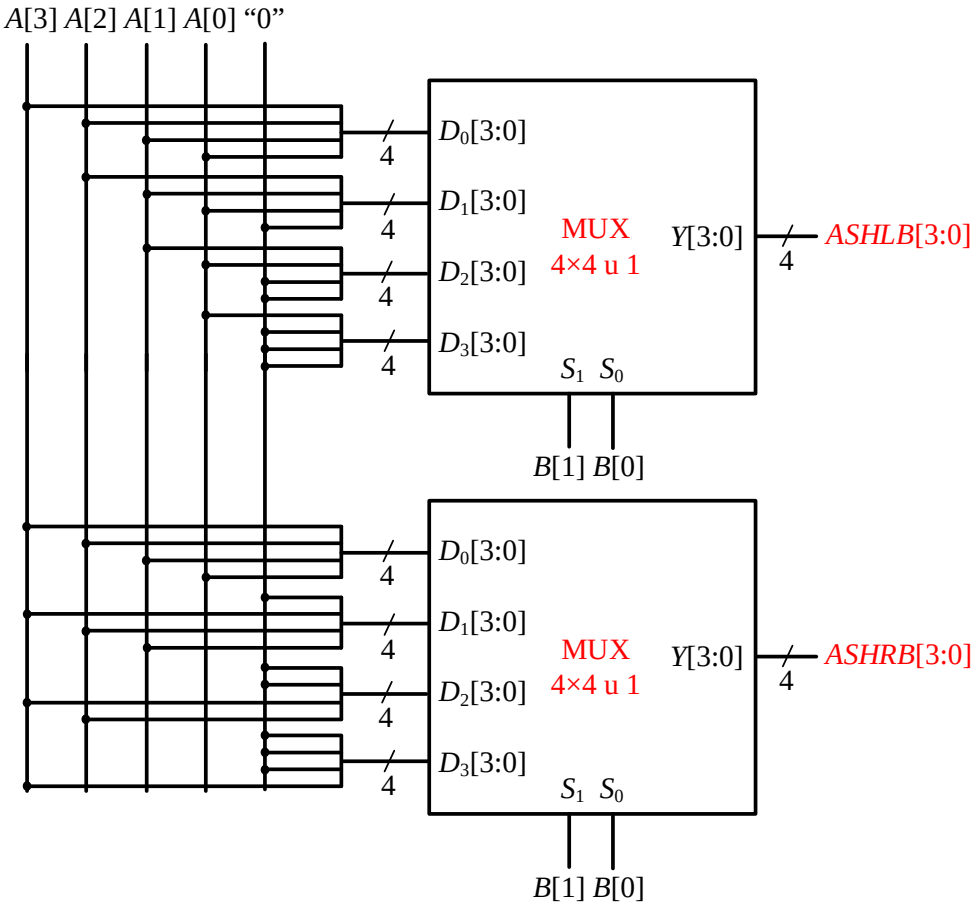
Preostalo je još da generišemo rezultate za operacije SHL (logičko pomeranje u levo) i SHR (logičko pomeranje u desno). U tekstu zadatka je rečeno da broj *B* u ovim operacijama ne može biti negativan niti veći od 3. Stoga je najlakše pripremiti rezultate za *B* = 0000, *B* = 0001, *B* = 0010 i *B* = 0011, a zatim propustiti ih kroz multiplekser 4×4 u 1 gde su selecioni signali biti *B*[1] i *B*[0]. Rezultati SHL i SHR instrukcije za različite vrednosti broja *B* su prikazani u tabeli 6.1, a primer u tabeli 6.2. Kombinaciona mreža koja realizuje operacije pomeranja je data na slici 6.3.

*Tabela* 6.1 – *Operacije logičkog pomeranja ulevo i udesno*

| Instrukcija/*B* | ***B* = 0000** | ***B* = 0001** | ***B* = 0010** | ***B* = 0011** |
|---|---|---|---|---|
| ***A* SHL *B*** | $A_3\ A_2\ A_1\ A_0$ | $A_2\ A_1\ A_0\ 0$ | $A_1\ A_0\ 0\ 0$ | $A_0\ 0\ 0\ 0$ |
| ***A* SHR *B*** | $A_3\ A_2\ A_1\ A_0$ | $0\ A_3\ A_2\ A_1$ | $0\ 0\ A_3\ A_2$ | $0\ 0\ 0\ A_3$ |

*Tabela* 6.2 – *Operacije logičkog pomeranja ulevo i udesno - primer*

| Instrukcija/*B* | ***B* = 0000** | ***B* = 0001** | ***B* = 0010** | ***B* = 0011** |
|---|---|---|---|---|
| ***A* SHL *B*** | 1011 | 0110 | 1100 | 1000 |
| ***A* SHR *B*** | 1011 | 0101 | 0010 | 0001 |

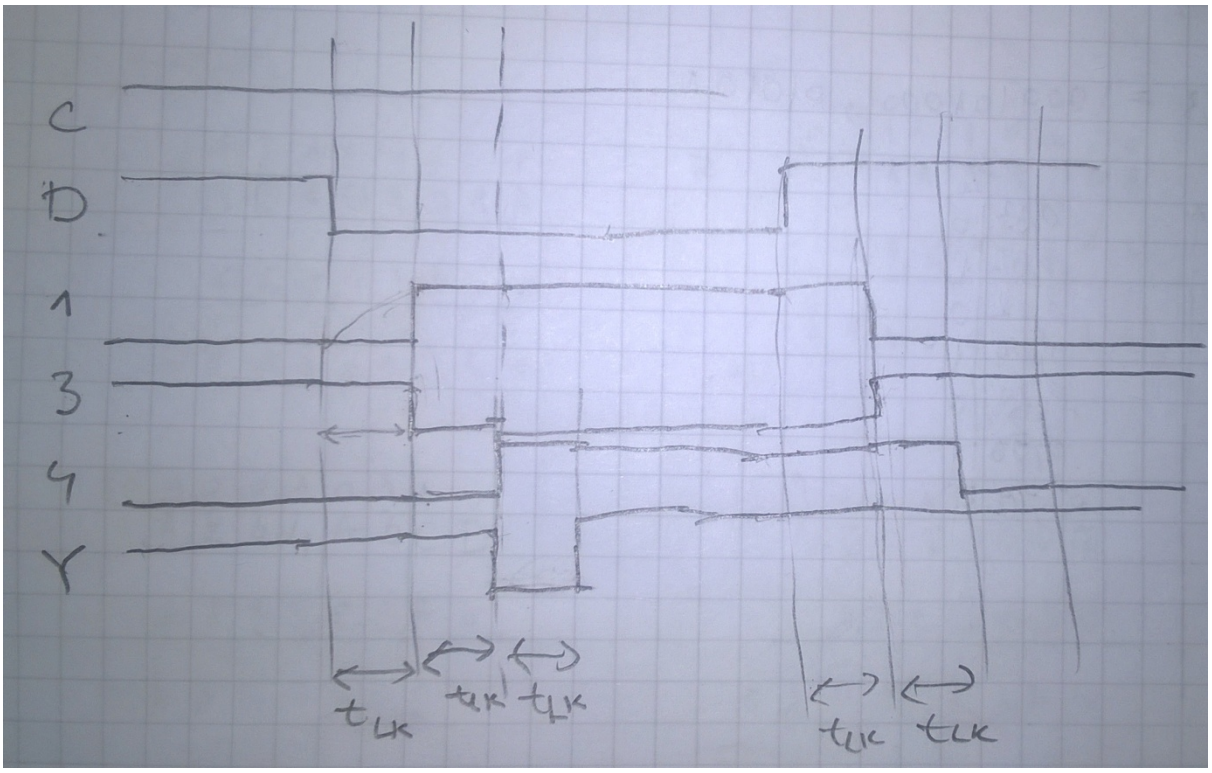



*Slika* 6.3 – *Operacije pomeranja ulevo i udesno*

---

Dodatni materijali za vežbanje se mogu naći u starim ispitnim rokovima i na sledećem linku:

- Dodatni zadaci iz kombinacionih mreža na ODE na elektronici , zadaci 1, 2, 3 (tačke a i b) i 4, kao i zadatak 9 koji je značajan za teorijski deo ispita
  Vremenski dijagrami za 9. zad:




- Dodatni zadaci iz kombinacionih mreža na ODE na elektronici 2 , zadaci 1, 2 i 6 (tačke a, b i c).