POGLAVLJE 3.
STANDARDNE KOMBINACIONE MREŽE

Na početku ovog poglavlja su date definicije standardnih kombinacionih mreža. Nakon toga se u zadacima ilustruju osnovni principi projektovanja minimalnih oblika standardnih kombinacionih mreža pomoću šema logičkih elemenata i VHDL jezika za opis fizičke arhitekture. Praktični primeri minimizacije Bulovih funkcija su prisutni u mnogim zadacima.

Analizom klase problema, koji se u složenim digitalnim sistemima mogu rešiti kombinacionim mrežama, uočeno je da postoje standardne jednoznačno definisane funkcije bez obzira na problem koji se rešava. To omogućava da se u projektovanju digitalnih sistema, kombinacione mreže koje im odgovaraju tretiraju preko zadate transformacije ulazne reči u izlaznu.

Uz primenu Bulove algebre i opisa transformacije ulazne reči u izlaznu pomoću Bulovih funkcija, teorijski je moguće projektovati proizvoljnu kombinacionu mrežu. Međutim, u slučaju projektovanja složenih sistema, taj metod nije praktičan, stoga što bi rezultantna mreža imala veliki broj logičkih kola, sa velikim brojem ulaza i složenim međuspajanjem. Kašnjenje signala kroz ovakvu mrežu takođe ne bi bilo zanemarivo.

Iz tog razloga se složene kombinacione mreže realizuju na modularan način, odnosno, dekompozicijom mreže u podmreže (module) koje se projektuju nezavisno jedna u odnosu na drugu. Moduli na koje se složena kombinaciona mreža dekomponuje mogu biti standardni ili namenski. U prvom slučaju radi se o modulima koji se koriste u raznovrsnim primenama, dok se u drugom slučaju radi o modulima koji su prilagođeni samo datom problemu. Uz pomoć namenskih modula se postiže bolja realizacija konkretnog problema sa stanovišta modularnosti i brzine.

Princip projektovanja modula standardnih kombinacionih mreža je omogućio da se složena kombinaciona mreža realizuje na regularan način preko skupa logičkih kola u kućištu integrisane komponente (MSI, SSI, VSI i VLSI skale integracije).

Modularan opis digitalnih sistema podrazumeva korišćenje hijerarhijskog sistema projektovanja. Pri tome, međusobno povezani moduli predstavljaju jedan hijerarhijski nivo. Međutim, jedan od modula sa ovog hijerarhijskog nivoa se može sastojati iz veze nekih drugih modula. U tom slučaju se dobija drugi hijerarhijski nivo. Ovakva dekompozicija modula digitalnog sistema se može uraditi sve do nivoa osnovnih elementarnih logičkih kola (I, ILI, NE), pri čemu dubina hijerarhije nije ograničena. To zapravo znači da sam projektant digitalnog sistema određuje dubinu hijerarhije dekompozicijom sistema na njegove sastavne module, sve do nivoa standarnih modula koji mu stoje na raspolaganju.

Ovakav pristup projektovanja kombinacionih mreža je omogućen i u VHDL jeziku za opis fizičke arhitekture. VHDL kao međunarodni standard za opis digitalnih sistema (prvi put predložen 1981. godine) omogućuje jednostavnu razmenu modela digitalnih sistema, čime inženjeri dobijaju jasan i jednoznačan uvid u funkcionalnost određenog modula digitalnog sistema.

Počevši od ovog poglavlja, VHDL jezik za opis fizičke arhitekture se koristi kao osnovni alat za rešavanje zadataka. Definicija VHDL jezika, njegovih osnovnih karakteristika i opis metodologije projektovanja koja se koristi nalazi se u devetom poglavlju knjige profesora Vladimira Kovačevića „Logičko projektovanje računarskih sistema I – Projektovanje digitalnih sistema".

Dekoder

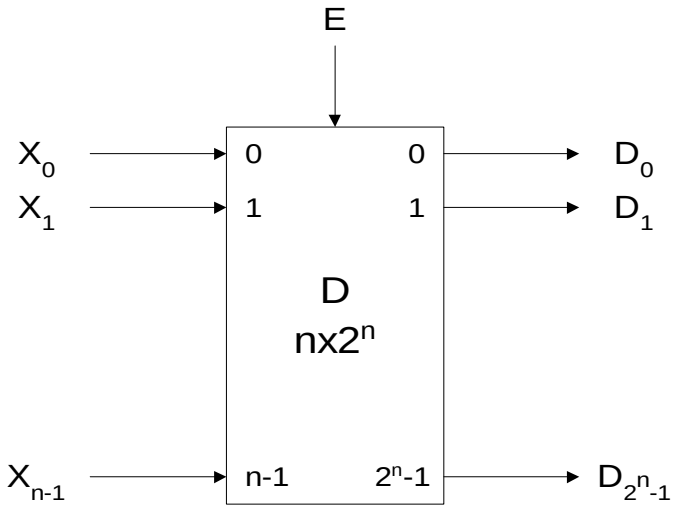
Dekoder je kombinaciona mreža koja poseduje n ulaznih i 2^n izlaznih priključaka. Mreža realizuje 2^n Bulovih funkcija.

$$D_j(X)=\begin{cases}1 & \text{za } x = j\\ 0 & \text{za } x \neq j\end{cases}$$

gde su j = 0, 1, 2, ... , 2^{n-1}; i $X=\sum_{i=1}^{n} x_i 2^{i-1}$ $\qquad x_i \in \{0,1\}$

Drugim rečima, dekoder transformiše n-elementarni binarni težinski kod u 2^n-elementarni kod, jer svakoj reči na ulazu dekodera odgovara jedinični signal samo na jednom izlaznom priključku.

Da bi se olakšalo korišćenje modula dekodera u projektovanju kombinacione mreže, dekoderu se dodaje dodatni ulaz dozvole (Slika 3.1).



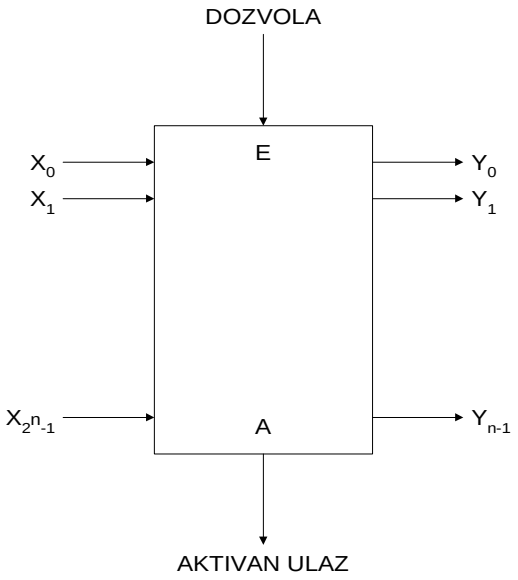
Slika 3.1: Simbol dekodera $n\times 2^n$

Kada je E=0 svi izlazi se nalaze na niskom nivou ili se nalaze u stanju visoke impedanse.

Koder

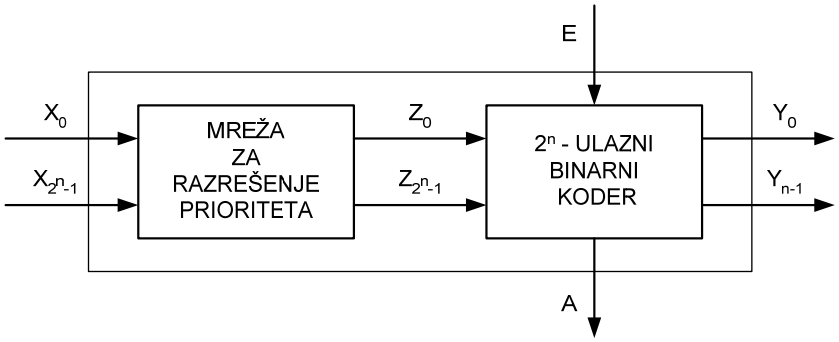
Koderi su kombinacione mreže sa 2^n ulaza (x_{2^n-1}, ..., x_0) informacije i n izlaza koji se nazivaju adresama.

Slika 3.2 prikazuje simbol uopštenog kodera koji ima 2^n ulaza (X_0 do X_{2^n-1}) i n izlaza (Y_0 do Y_{n-1}). sa ulazom dozvole (E) i izlazom koji ukazuje da je ulaz aktivan (A).



Slika 3.2: Simbol kodera

Ovakva realizacija kodera ima jedan nedostatak: nije utvrđen prioritet ulaza, tj. vrednost izlaznih signala u slučaju kad je istovremeno aktivno više ulaza. Problem utvrđivanja prioriteta ulaza rešava se tzv. prioritetnim koderima. Kod prioritetnih kodera u slučaju istovremene aktivnosti dva ili više ulaza, na izlazu se koduje onaj čiji je "redni broj" viši. U realizaciji prioritetnog kodera moguće je uočiti dva modula: prvi je kombinaciona mreža koja razrešuje prioritet ulaza, a drugi je "običan" koder. Moduli su povezani tako da se izlazi prvog dovode na ulazne priključke drugog (Slika 3.3).



Slika 3.3: Prioritetni koder

Mreža za razrešenje prioriteta eliminiše sve jedinice, izuzev one najvišeg prioriteta, dok je druga mreža binarni dekoder.

Mreža za razrešenje prioriteta je kombinaciona mreža sa 2^n (X_0 do X_{2^n-1}) ulaza i 2^n izlaza (Z_0 do Z_{2^n-1}) pri čemu je:

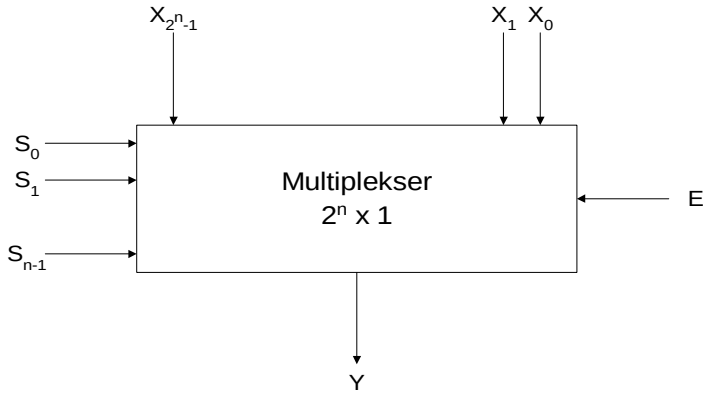
$$Z_i = \begin{cases} 1 & \text{za } x_i = 1 \text{ i } x_k = 0 \text{ za } k > i \\ 0 & \text{u protivnom} \end{cases}$$

Mreža za razrešenje prioriteta 4 ulaza se realizuje u VHDLu na sledeći način:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY test IS PORT (
  X: IN  std_logic_vector(3 DOWNTO 0);
  Z: OUT std_logic_vector(3 DOWNTO 0) );
END test;

ARCHITECTURE ath_test OF test IS
BEGIN
  Z(3) <= X(3);
  Z(2) <= NOT(X(3)) AND X(2);
  Z(1) <= NOT(X(3)) AND NOT(X(2)) AND X(1);
  Z(0) <= NOT(X(3)) AND NOT(X(2)) AND NOT(X(1)) AND X(0);
END arh_test;
```

Multiplekser

Multiplekseri su kombinacione mreže sa 2^n ulaznih priključaka ($x_{2^n-1},...,x_0$), n adresnih ($s_{n-1},...,s_0$), jednim izlaznim priključkom i ulazom dozvole E, Slika 3.4.



Slika 3.4: Simbol multipleksera

Vrednost adresnih promenljivih S se interpretira kao ceo broj u opsegu od 0 do 2^{n-1}, izražen kao binarni kod. Drugim rečima, ukoliko se na adresne priključke dovede adresa j, na izlaznom priključku će se pojaviti ulazni signal ranga j. U tom smislu miltiplekser služi kao višekanalni prekidač koji selektuje ulazne kanale na osnovu stanja adresnih ulaza.

Multiplekser se analitički definiše u obliku:

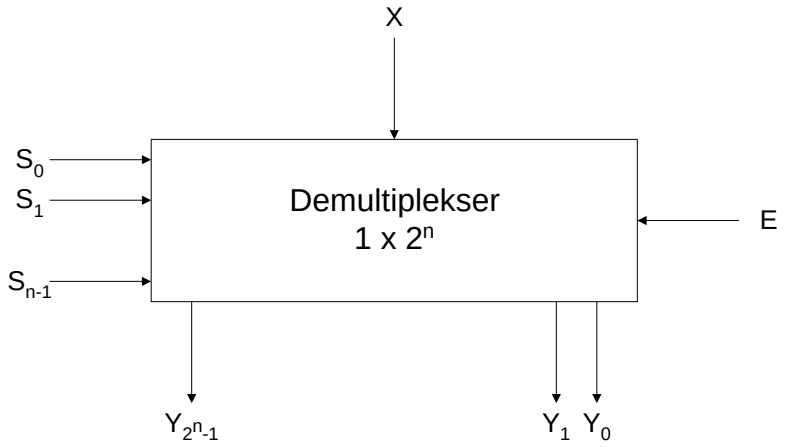
$$Y=\begin{cases} x_s & \text{ako je } E=1 \\ 0 & \text{ako je } E=0 \end{cases}$$

gde je $s=\sum_{j=0}^{n-1} s_j 2^j$ odnosno $Y=E\cdot\sum_{i=1}^{n} x_i p_i(s)$ gde su p_i potpuni proizvodi (konstituente jedinice) adresnih promenljivih.

Demultiplekser

Demultiplekser je kombinaciona mreža sa 2^n izlaza $Y = (Y_{2^n-1}, \ldots, Y_0)$, n adresnih (selekcionih) ulaza $S = (S_{n-1}, \ldots, S_0)$, jednim ulazom podataka X i priključkom za dozvolu demultipleksiranja E (Slika 3.5).

Ova mreža obavlja inverznu operaciju u odnosu na multiplekser, odnosno, ulaz X se usmerava na izlazni priključak Y izabran dovedenom adresom na adresni priključak S.



Slika 3.5: Blok šema demultipleksera

Demultiplekser se analitički definiše u obliku:

$$Y_i = \begin{cases} x_S & \text{ako je } i = S \text{ i } E = 1 \\ 0 & \text{ako je } i \neq S \text{ i } E = 0 \end{cases}$$

gde je $S = \sum_{j=0}^{n-1} s_j 2^j \quad 0 \le i \le 2^n - 1$ uzimajući u obzir ulaz dozvole E izlaz se može napisati u obliku $Y_i = E \cdot x \cdot p_i$ gde je p_i konstituenta jedinice adresnih ulaza.

Pomerači

Prost pomerač je kombinaciona mreža sa (n+2) ulaza X, n izlaza Y, dva upravljačka ulaza (za definisanje smera pomeranja (d) i definisanje da li ima pomeranja ili nema (s)).

U cilju realizacije mreža pomerača, uvodi se upravljački priključak dozvole ulaza E (y=0 za E=0).

Izlaz iz pomerača odgovara ulazu $X=(X_n, X_{n-1}, \ldots, X_0, X_{-1})$ pomerenom za jedan bit u levo ili desno, ili nepromenjenom ulazu, kako je to definisano priključkom smera d, odnosno, ulazom s. Rad pomerača se definiše na sledeći način:

$$Y_i = \begin{cases} X_{i-1} & \text{za } d=1 \wedge s=1 \wedge E=1 \quad \text{(levi pomeraj)} \\ X_{i+1} & \text{za } d=0 \wedge s=1 \wedge E=1 \text{ (desni pomeraj)} \\ X_j & \text{za } s=0 \wedge E=1 \quad \text{(nema pomeraja)} \\ 0 & \text{za } E=0 \end{cases}$$

gde je $0 \le i \le n-1$

Razlikuju se dve vrste pomeranja: logičko i aritmetičko. Princip pomeranja je isti kod obe vrste pomeranja, sa tom razlikom da se kod aritmetičkog pomeranja vodi računa o znaku ulaznog sloga. Tačnije rečeno, aritmetičko pomeranje ne menja znak sloga koji se pomera, dok logičko pomeranje ne vodi računa o znaku.

- **Logičko pomeranje**

Kod logičkog pomeranja se razlikuju dve vrste pomeranja:

- pomeranje sa ubacivanjem nule i
- pomeranje sa ubacivanjem jedinice

Princip kod obe vrste pomeranja je isti, sa razlikom u vrednosti bita koji se smešta (ubacuje) na upražnjeno mesto. To će se ilustrovati na primeru pomeranja četvorobitne vrednosti 0110:

	pomeranje sa ubacivanjem nule - 0	pomeranje sa ubacivanjem jedinice - 1
levi pomeraj: shl(0110) =	1100	1101
desni pomeraj: shr(0110) =	0011	1011

Uopšteni prikaz logičkog pomeranja prikazuje Tabela 3.1.

slog koji se pomera	b_3	b_2	b_1	b_0

shl	b_2	b_1	b_0	0 (1)
shr	0 (1)	b_3	b_2	b_1

Tabela 3.1: Ilustracija logičkog pomeranja

- **Aritmetičko pomeranje**

Karakteristika aritmetičkog pomeranja je da očuvava znak sloga koji se pomera. Pri tome je levi pomeraj za jednu poziciju jednak množenju sa 2, a desni pomeraj za jednu poziju predstavlja deljenje sa 2. Znak sloga se nalazi na bitu najveće važnosti (MSB bit) za sva tri načina predstavljanja brojeva:

- Znak + moduo
- I komplement
- II komplement

Način aritmetičkog pomeranja u levo će se ilustrovati na primeru broja 3, dok će se desni aritmetički pomeraj ilustrovati na primeru broja 6, za sve tri predstave brojeva.

Znak + moduo

Na ovaj način se broj 3 binarno predstavlja kao 0011, dok je njegova negativna vrednost 1011. Slično tome broj 6 je 0110, a –6 se predstavlja sa 1110.

Pomeranjem ulevo broja 3 za jednu poziciju treba da dobijemo 6. Na osnovu toga se zaključuje da se kod pomeranja ulevo na upražnjeno mesto bita najniže važnosti (LSB bit) ubacuje vrednost nula. Pri tome bit najveće važnosti (znak) ostaje na svom mestu i vrši se pomeranje samo preostalih bita. Isti ovaj zaključak važi i za negativne brojeve. To se ilustrije na sledeći način:

levi pomeraj: shl(0011) = 0110
levi pomeraj: shl(1011) = 1110

Pomeranje udesno broja 6 za jednu poziciju treba da za rezultat da broj 3. Slično je i kod negativnih brojeva gde pomeranjem udesno broja –6 treba da se dobije –3 (deljenje sa dva):

desni pomeraj: shr(0110) = 0011
desni pomeraj: shr(1110) = 1011

Vidi se da pri pomeranju udesno znak takođe ostaje na svom mestu, dok se preostali biti pomere udesno i da se na prazno mesto, na MSB-1 poziciji, ubacuje 0.

I komplement

Na ovaj način se broj 3 binarno predstavlja kao 0011, dok je njegova negativna vrednost 1100. Slično tome broj 6 je 0110, a –6 se predstavlja sa 1001.

Pomeranjem ulevo broja 3 treba da se dobije broj 6. Takođe, pomeranje ulevo broja –3 treba da rezultuje sa –6 zbog toga što je pomeranje ulevo ekvivalentno sa množenjem sa 2. Sledi ilustracija ovih rezultatata:

levi pomeraj: shl(0011) = 0110
levi pomeraj: shl(1100) = 1001

Analizom rezultata može se zaključiti da se u slučaju pozitivnih brojeva na upražnjeno mesto ubacije 0, dok se u slučaju negativnih brojeva ubacuje 1. Primećuje se da je bit koji se ubacuje zapravo jednak bitu znaka broja koji se pomera.

Kod pomeranja udesno važi isto pravilo, tj. da se na upražnjeno mesto ubacuje bit znaka. To će se ilustrovati na deljenju broja 6 (-6), gde treba za rezultat da se dobije 3 (-3):

desni pomeraj: shr(0110) = 0011
desni pomeraj: shr(1001) = 1100

II komplement

Predstavljanje brojeva pomoću II komplementa je najčešća metoda koja se koristi u digitalnim sistema. Na ovaj način se broj 3 binarno predstavlja kao 0011, dok je njegova negativna vrednost 1101. Slično tome broj 6 je 0110, a –6 se predstavlja sa 1010.

Pomeranje u levo i u ovoj predstavi brojeva predstavlja množenje sa dva, tako da se pomeranjem u levo broja 3 (-3) dobija broj 6 (-6). U nastavku se prikazuje ilustracija pomeranja u levo u predstavi brojeva pomoću II komplementa:

levi pomeraj: shl(0011) = 0110
levi pomeraj: shl(1101) = 1010

Primećuje se da i za pozitivne i za negativne brojeve na upražnjeno mesto na LSB poziciju prilikom pomeranja u levo ubacije vrednost 0.

Pomeranje u desno je isto kao i u predstavi pomoću I komplementa. Odnosno, na upražnjeno mesto na MSB-1 poziciji se ubacuje bit znaka i za pozitivne i za negativne brojeve:

desni pomeraj: shr(0110) = 0011
desni pomeraj: shr(1010) = 1101

Sumiranje svih prethodno navedenih pomeranja prikazuje Tabela 3.2.

slog koji se pomera	znak	b_2	b_1	b_0

- **znak+moduo**

shl	znak	b_1	b_0	0
shr	znak	0	b_2	b_1

- **I komplement**

shl	znak	b_1	b_0	znak
shr	znak	znak	b_2	b_1

- **II komplement**

shl	znak	b_1	b_0	0
shr	znak	znak	b_2	b_1

Tabela 3.2: Ilustracija aritmetičkog pomeranja

Komparator

Upoređivanje brojeva u digitalnim računarima vrši se u obliku: upoređivanja po modulu, upoređivanja znakova operanata i upoređenja redosleda. Najpotpunija operacija upoređivanja se odnosi na ispitivanje ispunjenja jednog od uslova:

$$A = B, \quad A > B, \quad A < B, \quad \text{gde su}$$

$$A = \pm \sum_{i=1}^{n} A_i 2^{i-1} \quad \text{i} \quad B = \pm \sum_{i=1}^{n} B_i 2^{i-1}$$

Postoje dva najčešća metoda za realizaciju operacije upoređenja. Kod prvog se od jednog broja oduzima drugi i prema znaku ostatka se donosi odluka o dva zadnja uslova. Ako je rezultat oduzimanja nula, tada je ispunjen prvi uslov. Za potrebe izvršavanja ove operacije je neophodan sabirač.

Da bi se ubrzala operacija upoređivanja i sabirač rasteretio za izvršavanje drugih operacija, koristi se drugi metod upoređivanja preko posebne kombinacione mreže upoređivanja.

Komparator modula (apsolutnih vrednosti) je kombinaciono kolo koje poredi dva broja A i B i određuje njihov odnos, pa su izlazi iz komparatora tri binarne promenljive koje određuju da li je A>B, A<B ili A=B.

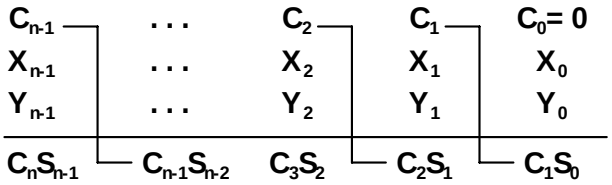
Sabirač

Sabirač predstavlja kombinacionu mrežu koja vrši aritmetičko sabiranje brojeva:

$$X = \sum_{i=1}^{n} x_i 2^{i-1} \quad \text{i} \quad Y = \sum_{i=1}^{n} y_i 2^{i-1}$$

$$x_i \in \{0,1\} \qquad y_i \in \{0,1\}$$

Proces sabiranja dva n-bitna binarna broja može se pokazati na sledeći način:



Očigledno je da su pri sabiranju mogući sledeći slučajevi:

$\left.\begin{matrix} S=1 \\ C=0 \end{matrix}\right\}$ - ako je samo po jedan sabirak jednak jedinici

$\left.\begin{matrix} S=0 \\ C=1 \end{matrix}\right\}$ - ako su po dva sabirka jednaka jedinici

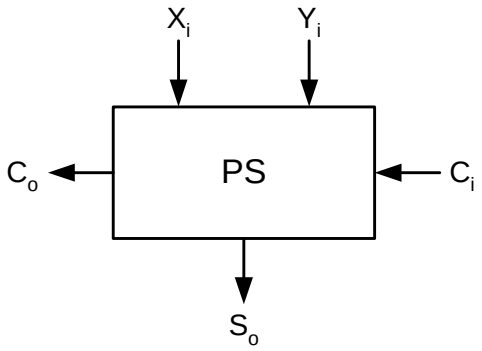
$\left.\begin{matrix} S=1 \\ C=1 \end{matrix}\right\}$ - ako su sva tri sabirka jednaka jedinici

Punim sabiračem naziva se kombinaciona mreža sa tri ulaza X_i, Y_i, C_i i dva izlaza S_i, C_{i+1}. koji predstavljaju vrednost zbira S_i i izlaznog prenosa C_{i+1}, jednog razreda binarnih brojeva X i Y. Prema tome, ulazi punog sabirača su (Slika 3.6):

- X_i i Y_i (kao biti koje treba sabrati) i
- C_i (prenos sa bita manje težine)

dok su izlazi punog sabirača

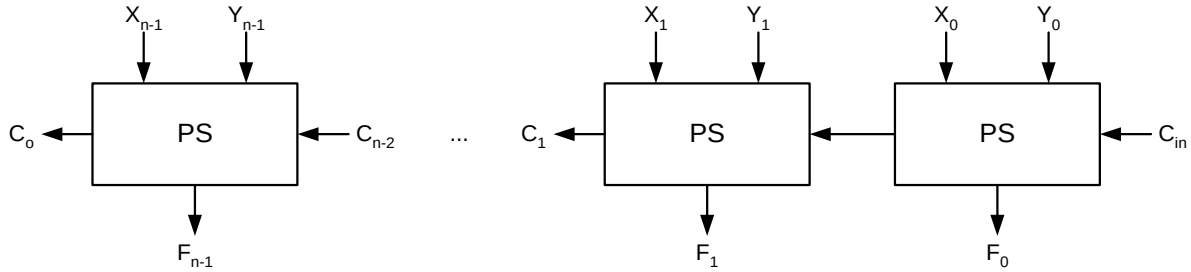
- S_o (vrednost zbira) i
- C_o (prenos za bit veće težine).



Slika 3.6: Ulazno/izlazni signali punog sabirača

Jedan od načina realizacije sabirača reda n je tzv. paralelni sabirač. Paralelni sabirač reda n se realizuje od n punih sabirača, povezanih tako da se operacije izvršavaju nad svim bitima operanada istovremeno (Slika 3.7).

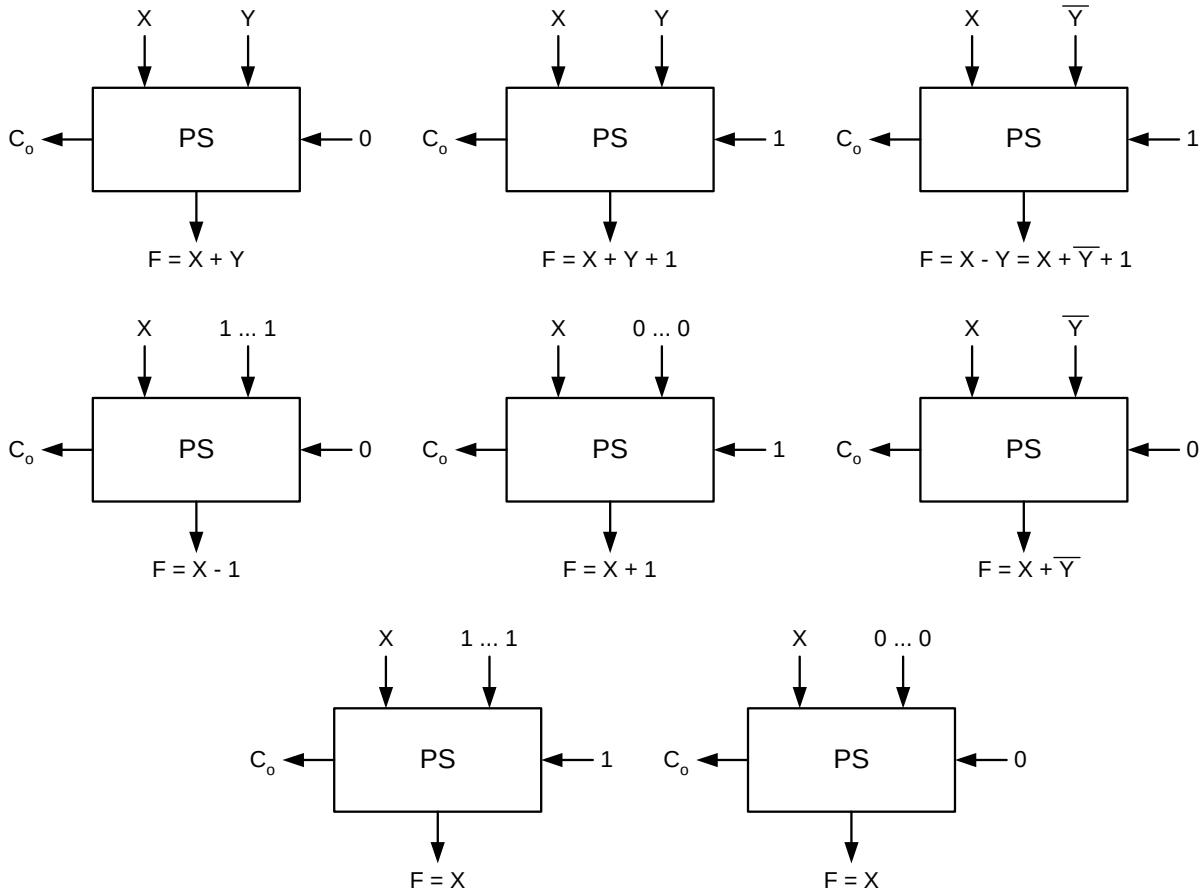
Rezultat aritmetičkih operacija nad bitima manje težine direkno utiče na vrednost bita veće težine preko signala izlaznog prenosa kao $C_{i(i)} = C_{o(i-1)}$, dok kod logičkih operacija ovog prenosa nema ($C_i = 0$).



Slika 3.7: Blok šema paralelnog sabirača sa n razreda

Niz različitih aritmetičkih i logičkih mikrooperacija moguće je ostvariti upravljanjem ulazima u paralelni sabirač. Slika 3.8 prikazuje realizaciju osam različitih operacija pomoću paralelnog sabirača.

Logičke operacije ostvaruju se zabranom svih ulaznih prenosa u pun sabirač i dovođenjem odgovarajućih oblika ulaza.



Slika 3.8: Aritmetičke operacije realizovane različitim upravljanjem ulazima paralelnog sabirača

3.1 ZADATAK:

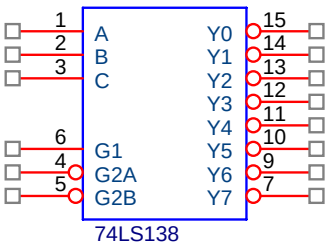
Integrisana komponenta 74LS138 predstavlja matrični dekoder 3×8. To je kombinaciona mreža koja jednu od osam izlaznih linija (Y_0 do Y_7) postavlja u aktivno stanje, u zavisnosti od stanja u kome se u tom trenutku nalaze tri binarna ulaza A, B i C, kao i ulazni signali dozvole ($G_1, \overline{G}_{2A}$ i $\overline{G}_{2B}$).

Tabela 3.3 prikazuje kombinacionu tabelu ulaza/izlaza ovog dekodera, gde je sa $\overline{G}_2^*$ predstavljen logički proizvod ulaznih signala dozvole $\overline{G}_{2A}$ i $\overline{G}_{2B}$, tj. $\overline{G}_2^* = \overline{G}_{2A} \cdot \overline{G}_{2B}$. Iz tabele se vidi da su izlazi ovog dekodera predstavljeni u negativnoj logici, odnosno izlaz Y_i je aktivan kada se nalazi u stanju logičke nule. Ulazi dekodera C, B i A su predstavljeni u pozitivnoj logici.

ULAZI DEKODERA					IZLAZI DEKODERA							
G_1	$\overline{G}_2^*$	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×	H	×	×	×	H	H	H	H	H	H	H	H
L	×	×	×	×	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Tabela 3.3: Kombinaciona tabela dekodera 74LS138

Slika 3.9 prikazuje simbol dekodera 74LS138.



Slika 3.9: Logički simbol DEKODERA 74LS138

Kada je $G_1 \cdot \overline{G}_{2A} \cdot \overline{G}_{2B} = 1$, dekoderska NI kola u zavisnosti od kombinacije ulaznih signala, postavljaju odgovarajući izlazni signal na (aktivnu) nulu. U suprotnom, kada je $G_1 \cdot \overline{G}_{2A} \cdot \overline{G}_{2B} = 0$, svi izlazi dekodera će biti neaktivni, bez obzira na kombinaciju ulaznih signala.

a) Izvršiti sintezu dekodera 74LS138 pomoću NI logičkih kola
b) Izvršiti sintezu dekodera 74LS138 pomoću VHDL jezika za opis fizičke arhitekture

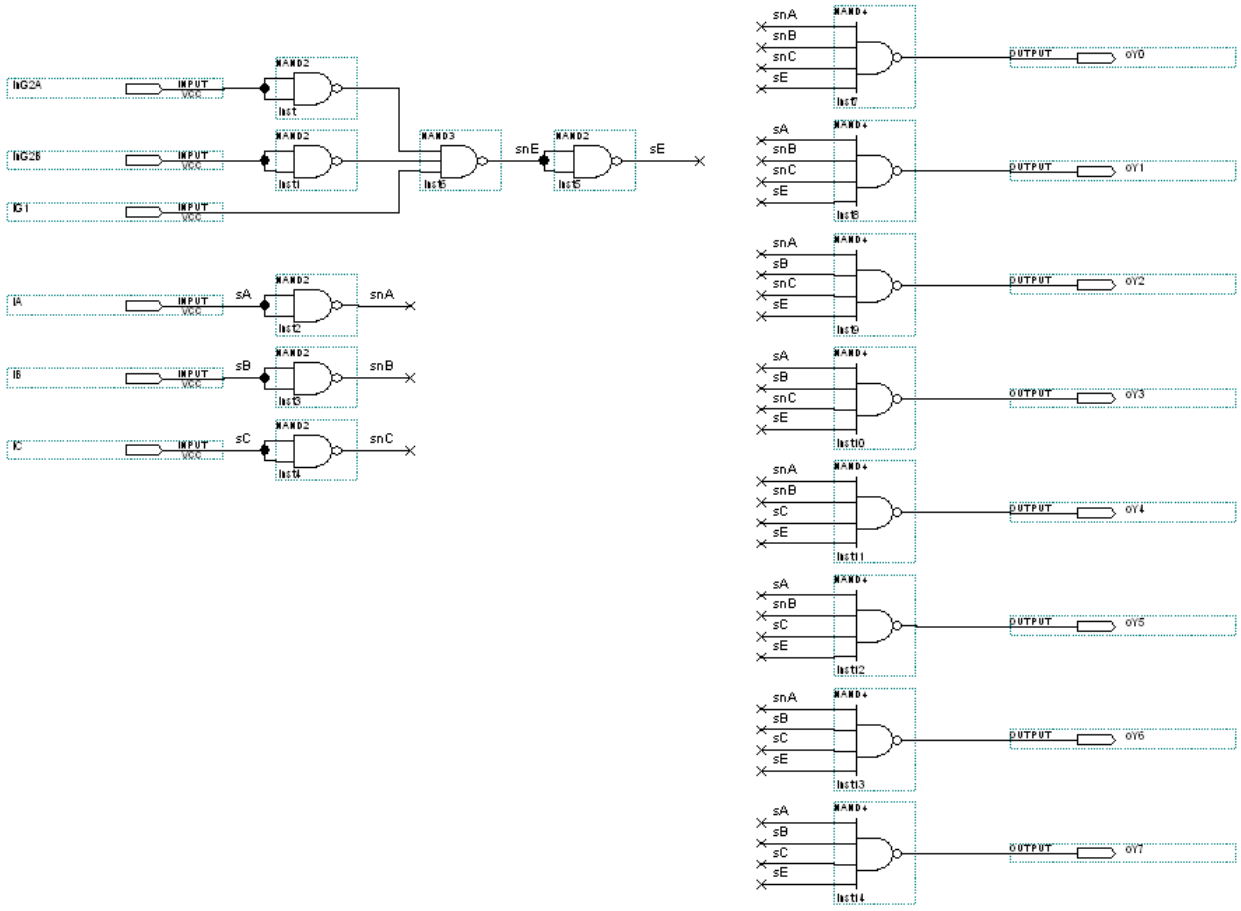
REŠENJE:

Funkcije pojedinih izlaza dobijamo direktno iz tabele, i to:

$$Y_0 = \overline{E \cdot \overline{C} \cdot \overline{B} \cdot \overline{A}}, \quad Y_1 = \overline{E \cdot \overline{C} \cdot \overline{B} \cdot A}, \quad Y_2 = \overline{E \cdot \overline{C} \cdot B \cdot \overline{A}}, \quad Y_3 = \overline{E \cdot \overline{C} \cdot B \cdot A},$$
$$Y_4 = \overline{E \cdot C \cdot \overline{B} \cdot \overline{A}}, \quad Y_5 = \overline{E \cdot C \cdot \overline{B} \cdot A}, \quad Y_6 = \overline{E \cdot C \cdot B \cdot \overline{A}}, \quad Y_7 = \overline{E \cdot C \cdot B \cdot A},$$

pri čemu oznaka E predstavlja dozvolu ulaza, $E = G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}$.

Slika 3.10 prikazuje logičku šemu realizacije matričnog dekodera 74LS138 pomoću NI logičkih kola.



Slika 3.10: Logička šema dekodera 74LS138

Pomoću VHDLa moguće je realizovati matrični dekoder na više načina. Jedan od opisa ove komponente je prikazan sledećim izvornim kodom:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY DECODER IS PORT(
  iA: IN  std_logic; -- ulazni signal na poziciji sa težinom 1
  iB: IN  std_logic; -- ulazni signal na poziciji sa težinom 2
  iC: IN  std_logic; -- ulazni signal na poziciji sa težinom 4
  iG1,
  inG2A,
```

```
    inG2B: IN  std_logic; -- ulazni signali dozvole dekodovanja
    oY: OUT std_logic_vector (7 DOWNTO 0) ); -- izlaz dekodera
END DECODER;

ARCHITECTURE ARH_DECODER OF DECODER IS
    -- vektor koji objedinjuje ulazne signale
    -- radi lakšeg dekodovanja
    SIGNAL sINPUT: std_logic_vector(2 DOWNTO 0);
BEGIN

    -- formiranje ulaznog vektora za dekoder
    sINPUT <= (iC & iB & iA);

    PROCESS (sINPUT, iG1, inG2A, inG2B) BEGIN
        -- provera zadovoljenja uslova dozvole dekodovanja
        IF (iG1 = '1' AND inG2A = '0' AND inG2B = '0') THEN
            -- dekodovanje je dozvoljeno -> formira se izlaz dekodera
            CASE sINPUT IS
                WHEN "000"  => oY <= "11111110";
                WHEN "001"  => oY <= "11111101";
                WHEN "010"  => oY <= "11111011";
                WHEN "011"  => oY <= "11110111";
                WHEN "100"  => oY <= "11101111";
                WHEN "101"  => oY <= "11011111";
                WHEN "110"  => oY <= "10111111";
                WHEN "111"  => oY <= "01111111";
                WHEN OTHERS => oY <= "11111111";
            END CASE;
        ELSE
            -- dekodovanje nije dozvoljeno ->
            -- svi izlazi dekodera su postavljeni na 1
            oY <= "11111111";
        END IF;
    END PROCESS;

END ARH_DECODER;
```

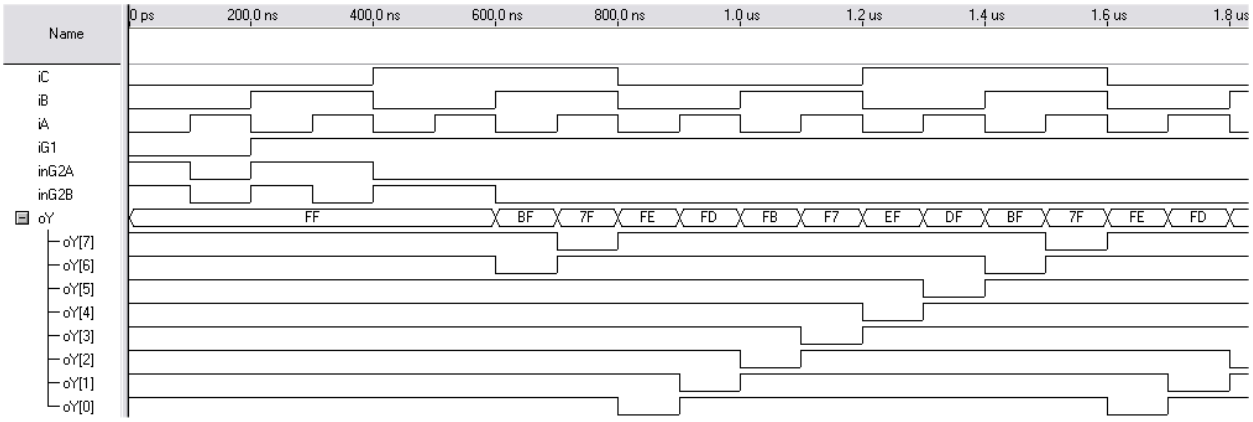
U okviru entiteta DECODER definisani su ulazno/izlazni signali dekodera:

1. Ulazni signali dekodera iA, iB i iC na osnovu kojih se aktivira odgovarajuća izlazna linija dekodera
2. Signali dozvole rada dekodera iG1, inG2A i inG2B, pri čemu su signali inG2A i inG2B aktivni na nivou logičke nule i signal iG1 je aktivan na nivou logičke jedinice
3. Izlaz dekodera je realizovan kao osmobitni vektor oY.

U okviru zaglavlja arhitekture je deklarisan trobitni vektor sINPUT kojem se u okviru tela arhitekture dodeljuje vrednost ulaznih signala dekodera iskazom sINPUT<=(iC & iB & iA). Arhitektutu ARH_DECODER pored navedenog iskaza čini i proces koji izvršava traženo dekodovanje. U listi osetljivosti ovog procesa su navedeni svi ulazni signali dekodera. U okviru procesa se na osnovu vrednosti vektora sINPUT, na izlazu postavlja kod aktivnog izlaznog signala u negativnoj logici, ukoliko su sva tri signala dozvole u aktivnom stanju, tj. iG1=1,

`inG2A=0` i `inG2B=0`. Za slučaj kada je barem jedan od tri signala dozvole u neaktivnom stanju, svi izlazi dekodera će biti neaktivni (`oY="11111111"`).

Slika 3.11 prikazuje vremenski dijagram rada realizovanog dekodera 3×8. Sa dijagrama se vidi da je izlaz dekodera validan kada su svi kontrolni signali u aktivnom stanju (`iG1=1`, `inG2A=0` i `inG2B=0`). Inače se na svim izlazima dekodera pojavljuju jedinice.



Slika 3.11: Vremenski dijagram dekododera 3×8

3.2 ZADATAK:

Izvršiti sintezu dekadnog BCD kodera (Tabela 3.4) pomoću logičkih kola i pomoću VHDL jezika za opis fizičke arhitekture.

U_0	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	D	C	B	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Tabela 3.4: Funkcionalna tabela dekadnog BCD kodera

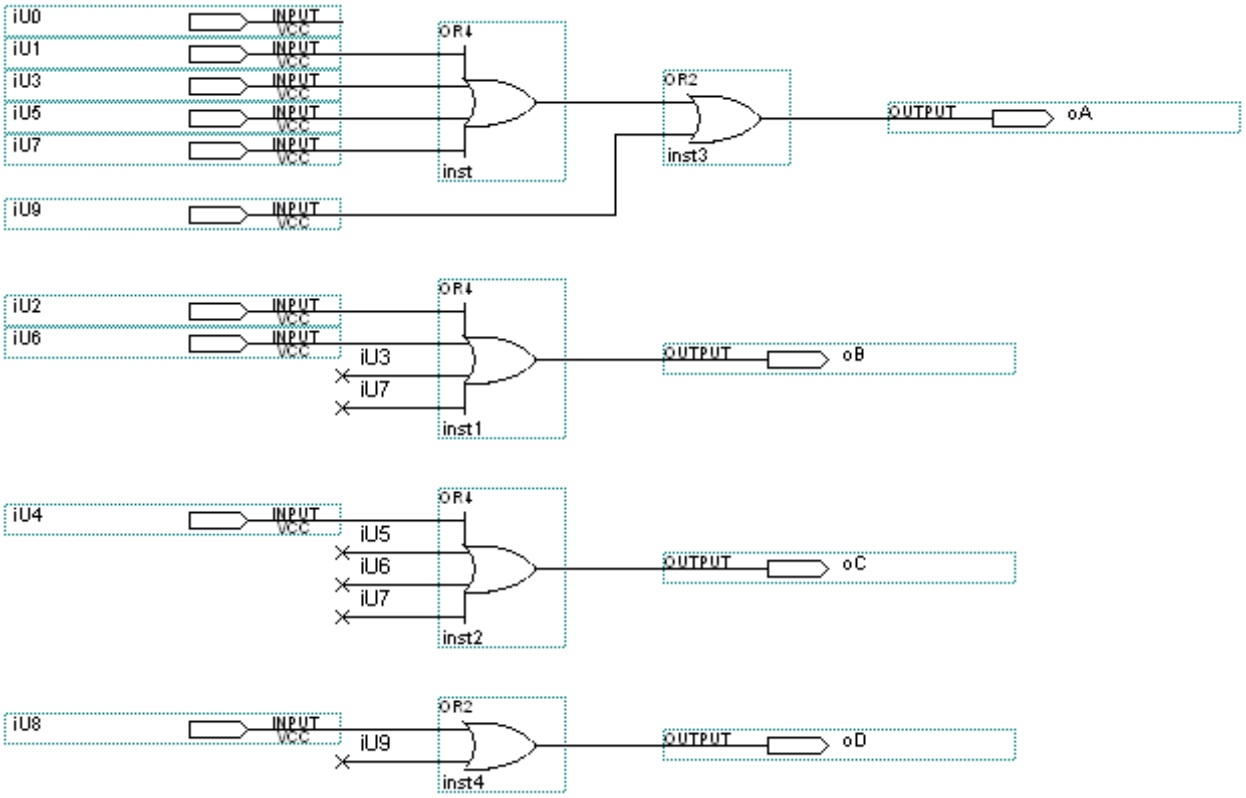
REŠENJE:

Jednostavnim iščitavanjem iz tabele dobijaju se Bulove jednačine za svaki od izlaza A, B, C i D:

$A = U_1 + U_3 + U_5 + U_7 + U_9$ $\qquad$ $B = U_2 + U_3 + U_6 + U_7$

$C = U_4 + U_5 + U_6 + U_7$ $\qquad$ $D = U_8 + U_9$

Na osnovu ovih jednačina može se formirati logička šema dekadnog BCD kodera (Slika 3.12).



Slika 3.12: Logička šema dekadnog BCD kodera

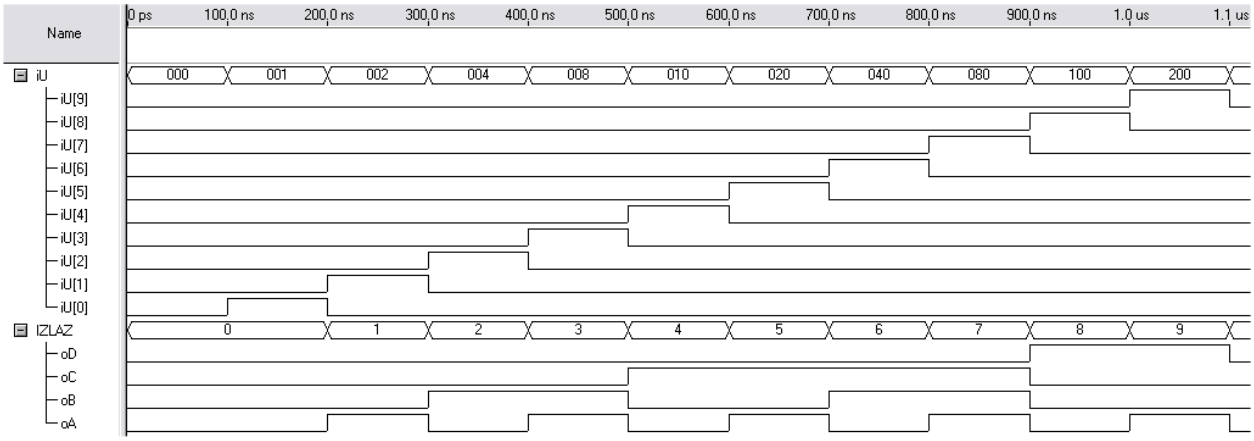
VHDL opis dekadnog BCD kodera je dat u nastavku teksta:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY BCD_KODER IS PORT (
  -- ulazni vektor
  iU:              IN  std_logic_vector(9 downto 0);
  -- izlazi dekadnog BCD kodera
  oA, oB, oC, oD: OUT std_logic );
END BCD_KODER;

ARCHITECTURE ARH_BCD_KODER OF BCD_KODER IS
  -- vektor koji prima odgovarajuću binarnu vrednost
  -- u zavisnosti od stanja ulaza kodera
  SIGNAL sBCD_VECTOR: std_logic_vector(3 downto 0);
BEGIN

  -- formiranje binarnog vektora
  PROCESS (iU) BEGIN
    sBCD_VECTOR <= "0000";
    IF (iU(0) = '1') THEN sBCD_VECTOR <= "0000"; END IF;
    IF (iU(1) = '1') THEN sBCD_VECTOR <= "0001"; END IF;
    IF (iU(2) = '1') THEN sBCD_VECTOR <= "0010"; END IF;
    IF (iU(3) = '1') THEN sBCD_VECTOR <= "0011"; END IF;
```

```
    IF (iU(4) = '1') THEN sBCD_VECTOR <= "0100"; END IF;
    IF (iU(5) = '1') THEN sBCD_VECTOR <= "0101"; END IF;
    IF (iU(6) = '1') THEN sBCD_VECTOR <= "0110"; END IF;
    IF (iU(7) = '1') THEN sBCD_VECTOR <= "0111"; END IF;
    IF (iU(8) = '1') THEN sBCD_VECTOR <= "1000"; END IF;
    IF (iU(9) = '1') THEN sBCD_VECTOR <= "1001"; END IF;
  END PROCESS;

  -- dodela vrednosti izlaznim signalima na osnovu
  -- formiranog binarnog vektora
  oD <= sBCD_VECTOR(3); -- bit sa težinom 8
  oC <= sBCD_VECTOR(2); -- bit sa težinom 4
  oB <= sBCD_VECTOR(1); -- bit sa težinom 2
  oA <= sBCD_VECTOR(0); -- bit sa težinom 1

END ARH_BCD_KODER;
```

Entitet `BCD_KODER` opisuje ulazno/izlazne signale dekadnog BCD kodera. Ulazni signali su predstavljeni vektorom `iU` koji obuhvata 10 signala (`9 downto 0`). Izlazni signali su označeni sa `oA`, `oB`, `oC` i `oD`, gde je `oA` bit najmanje važnosti (LSB), a `oD` bit najveće važnosti (MSB) u izlaznoj reči. Unutar arhitekture `ARH_BCD_KODER` definisan je pomoćni signal `sBCD_VECTOR`, kojem se na osnovu vrednosti ulaznih signala dodeljuje odgovarajuća binarna vrednost (na primer, ako je aktivan ulaz `iU(5)`, signal `sBCD_VECTOR` dobija vrednost `"0101"`). Nakon izvršenja procesa kod signal `sBCD_VECTOR` će imati odgovarajuću vrednost koja se posle preslikava na izlane signale `oA`, `oB`, `oC` i `oD`.

Slika 3.13 prikazuje vremenske dijagrame rada realizovanog dekadnog BCD kodera.



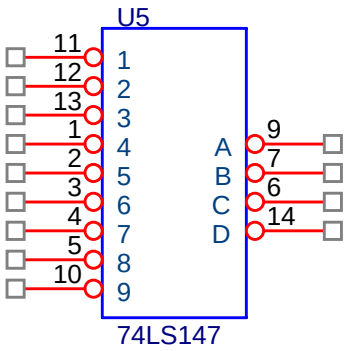
Slika 3.13: Vremenski dijagram dekadnog BCD kodera

Slika 3.13 prikazuje jedan veliki nedostatak ovakve realizacije kodera. Naime, izlazi BCD kodera isti su u slučaju kad nije aktivan ni jedan ulaz kao i kad je aktivan samo ulaz U_0. Ovaj problem se prevazilazi definisanjem dodatnog izlaza koji ima vrednost jedan ako je aktivan bar jedan od ulaza, a u suprotnom je jednak nuli. Pomenuti signal se može realizovati kao izlaz ILI kola na čije se ulaze dovode svi ulazni signali kodera. Drugi način rešenja ovog problema je uvođenje dodatnog

ulaza koji ukazuje da su ulazni signali za kodovanje validni. To znači da ako je ulaz dozvole na primer na niskom nivou tada na izlazu dekodera treba da bude validan kod ulaznih signala. Inače na izlaznim signalima treba da bude omogućena detekcija da nisu kodovani ulazni signali (jedno rešenje je postavljanje svih izlaznih signala na logičku jedinicu).

3.3 ZADATAK:

Integrisana komponenta 74LS147, Slika 3.14, predstavlja prioritetni koder. Tabela 3.5 prikazuje funkcionalnu zavisnost izlaza ove komponente u zavisnosti od njenih ulaza. Iz tabele se vidi se da su ulazi aktivni kada se nalaze na niskom nivou (logička 0). Izlazi su takođe predstavljeni u negativnoj logici.



Slika 3.14: Logički simbol prioritetnog kodera 74LS147

U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
×	×	×	×	×	×	×	×	L	L	H	H	L
×	×	×	×	×	×	×	L	H	L	H	H	H
×	×	×	×	×	×	L	H	H	H	L	L	L
×	×	×	×	×	L	H	H	H	H	L	L	H
×	×	×	×	L	H	H	H	H	H	L	H	L
×	×	×	L	H	H	H	H	H	H	L	H	H
×	×	L	H	H	H	H	H	H	H	H	L	L
×	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

Tabela 3.5: Funkcionalna tabela prioritetnog kodera 74LS147

a) Izvršiti sintezu prioritetnog kodera 74LS147 pomoću logičkih kola
b) Izvršiti sintezu prioritetnog kodera 74LS147 pomoću VHDL jezika za opis fizičke arhitekture

REŠENJE:

Na osnovu tabele dobijaju se sledeće jednačine izlaza:

$$D = \overline{\overline{U_9} + \overline{U_8} \cdot U_9}$$

$$C = \overline{\overline{U_7} \cdot U_8 \cdot U_9 + \overline{U_6} \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_5} \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_4} \cdot U_5 \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9}$$

$$B = \overline{\overline{U_7} \cdot U_8 \cdot U_9 + \overline{U_6} \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_3} \cdot U_4 \cdot U_5 \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_2} \cdot U_3 \cdot U_4 \cdot U_5 \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9}$$

$$A = \overline{\overline{U_9} + \overline{U_7} \cdot U_8 \cdot U_9 + \overline{U_5} \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_3} \cdot U_4 \cdot U_5 \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9 + \overline{U_1} \cdot U_2 \cdot U_3 \cdot U_4 \cdot U_5 \cdot U_6 \cdot U_7 \cdot U_8 \cdot U_9}$$

Primenom teoreme Bulove algebre koja se naziva apsorpcija:

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

izlaz D postaje $D = \overline{\overline{U_9} + \overline{U_8} \cdot U_9} = \overline{\overline{U_9} + \overline{U_8}} = U_9 \cdot U_8$.

Analogno, primenom prethodno prikazanog pravila, kao i asocijativnog i De-Morganovog zakona moguće je minimizovati i preostale izraze:

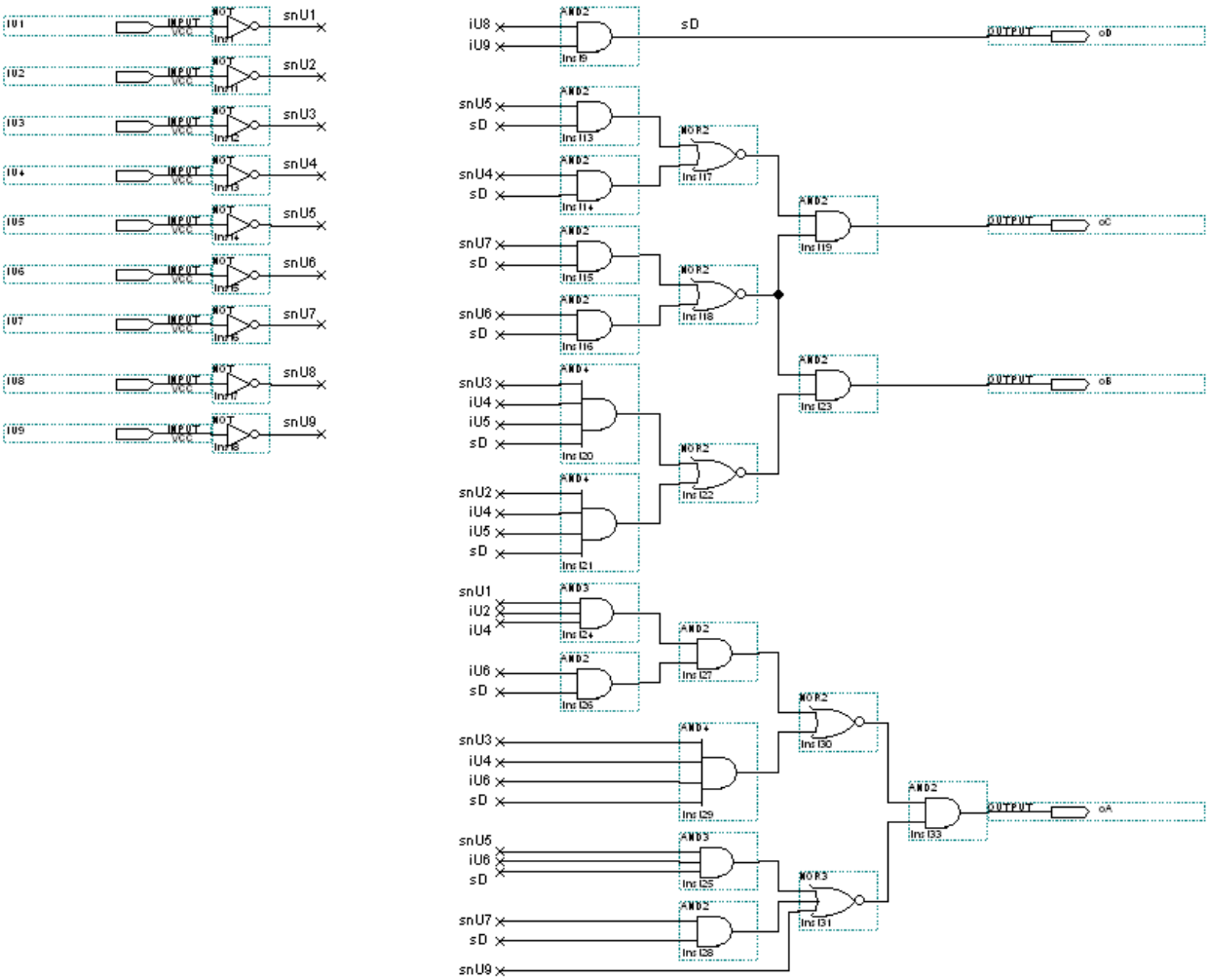
$$D = U_8 \cdot U_9$$

$$C = \overline{\overline{U_7} \cdot D + \overline{U_6} \cdot D} \cdot \overline{\overline{U_5} \cdot D + \overline{U_4} \cdot D}$$

$$B = \overline{\overline{U_7} \cdot D + \overline{U_6} \cdot D} \cdot \overline{\overline{U_3} \cdot U_4 \cdot U_5 \cdot D + \overline{U_2} \cdot U_4 \cdot U_5 \cdot D}$$

$$A = \overline{\overline{U_9} + \overline{U_7} \cdot D + \overline{U_5} \cdot U_6 \cdot D} \cdot \overline{\overline{U_3} \cdot U_4 \cdot U_6 \cdot D + (\overline{U_1} \cdot U_2 \cdot U_4) \cdot (U_6 \cdot D)}$$

Nakon izvršene minimizacije izlaznih signala prioritetnog kodera može se projektovati odgovarajuća logička šema (Slika 3.15).



Slika 3.15: Logička šema prioritetnog kodera 74LS147

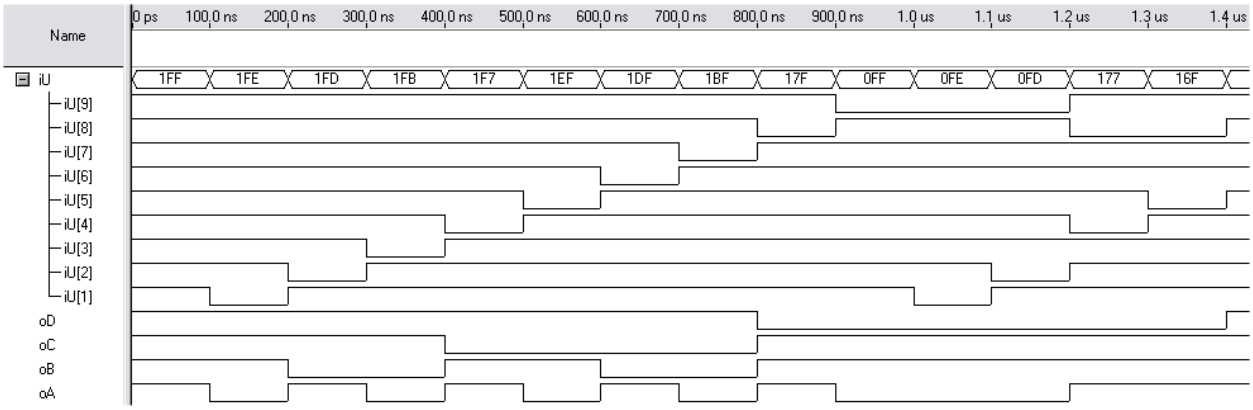
Sledi jedan način opisa prioritetnog BCD kodera uVHDLu:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY PRIORITY_CODER IS PORT (
  -- ulazni vektor
  iU:             IN  std_logic_vector(9 DOWNTO 1);
  -- izlaz prioritetnog BCD kodera
  oA, oB, oC, oD: OUT std_logic );
END PRIORITY_CODER;

ARCHITECTURE ARH_PRIORITY_CODER OF PRIORITY_CODER IS
  -- vektor koji prima odgovarajuću binarnu vrednost
  -- u zavisnosti od stanja ulaza kodera
  SIGNAL sBCD_VECTOR:std_logic_vector(3 DOWNTO 0);
BEGIN

  -- formiranje izlaznog vektora prema zadatim prioritetima
  PROCESS (iU) BEGIN
    IF    (iU(9)='0') THEN sBCD_VECTOR <= "0110";
    ELSIF (iU(8)='0') THEN sBCD_VECTOR <= "0111";
    ELSIF (iU(7)='0') THEN sBCD_VECTOR <= "1000";
    ELSIF (iU(6)='0') THEN sBCD_VECTOR <= "1001";
    ELSIF (iU(5)='0') THEN sBCD_VECTOR <= "1010";
    ELSIF (iU(4)='0') THEN sBCD_VECTOR <= "1011";
    ELSIF (iU(3)='0') THEN sBCD_VECTOR <= "1100";
    ELSIF (iU(2)='0') THEN sBCD_VECTOR <= "1101";
    ELSIF (iU(1)='0') THEN sBCD_VECTOR <= "1110";
    ELSE                   sBCD_VECTOR <= "1111";
    END IF;
  END PROCESS;

  -- dodela vrednosti izlaznim signalima na osnovu
  -- formiranog binarnog vektora
  oD <= sBCD_VECTOR(3); -- bit sa težinom 8
  oC <= sBCD_VECTOR(2); -- bit sa težinom 4
  oB <= sBCD_VECTOR(1); -- bit sa težinom 2
  oA <= sBCD_VECTOR(0); -- bit sa težinom 1

END ARH_PRIORITY_CODER;
```

Entitet `PRIORITY_CODER` opisuje ulazno izlazne signale dekadnog BCD kodera sa prioritetima. Ulazni signali su predstavljeni vektorom `iU` koji obuhvata 9 signala (`9 downto 1`). Izlazni signali su označeni sa `oA`, `oB`, `oC` i `oD`, gde je `oA` bit najmanje važnosti (LSB), a `oD` bit najveće važnosti (MSB) u izlaznoj reči. Unutar arhitekture `ARH_PRIORITY_CODER` definisan je pomoćni signal `sBCD_VECTOR`, kojem se unutar realizovanog procesa, na osnovu vrednosti ulaznih signala i u skladu sa dodeljenim prioritetima dodeljuje odgovarajuća binarna vrednost. Razrešenje problema prioriteta je u ovom rešenju realizovano sa jednim uslovnim iskazom `IF-THEN-ELSIF-ELSE`, tako da se na početku iskaza prvo proverava stanje signala sa najvišim prioritetom. Ako je taj signal aktivan pomoćnom signalu se daje odgovarajući kod. U suprotnom slučaju se prelazi na ispitivanje stanja signala sa stepenom prioriteta manjim za jedan, i tako redom do poslednjeg ulaznog signala. Ako nijedan ulazni signal nije aktivan pomoćna

promenljiva dobija vrednost F heksadecimalno. Nakon izvršenja procesa kod signal `sBCD_VECTOR` će imati odgovarajuću vrednost koja se posle preslikava na izlane signale `oA`, `oB`, `oC` i `oD`.

Slika 3.16 prikazuje vremenske dijagrame rada prioritetnog dekadnog BCD kodera.



Slika 3.16: Vremenski dijagram prioritetnog dekadnog BCD kodera

Na početku vremenskog dijagrama rada prioritetnog dekadnog BCD kodera su prikazane situacije kada je aktivan samo jedan ulaz. U drugom delu vremenskog dijagrama su prikazane situacije kada je aktivno više različitih ulaznih signala. Vidi se da je razrešenje prioriteta realizovano ispravno, tj. na izlazu se pojavaljuje kod signala sa višim prioritetom.

3.4 ZADATAK:

Realizovati multiplekser sa osam ulaza (X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0), tri adresna ulaza (S_2, S_1, S_0) i jednim izlazom (Y) pomoću logičkih kola i pomoću VHDL jezika za opis fizičke arhitekture. Tabelu ulaza/izlaza prikazuje Tabela 3.6.

E	S_2	S_1	S_0	Y
0	×	×	×	0
1	0	0	0	X_0
1	0	0	1	X_1
1	0	1	0	X_2
1	0	1	1	X_3
1	1	0	0	X_4
1	1	0	1	X_5
1	1	1	0	X_6
1	1	1	1	X_7

Tabela 3.6: Tabela kojom se opisuje funkcionisanje multipleksera 8x1

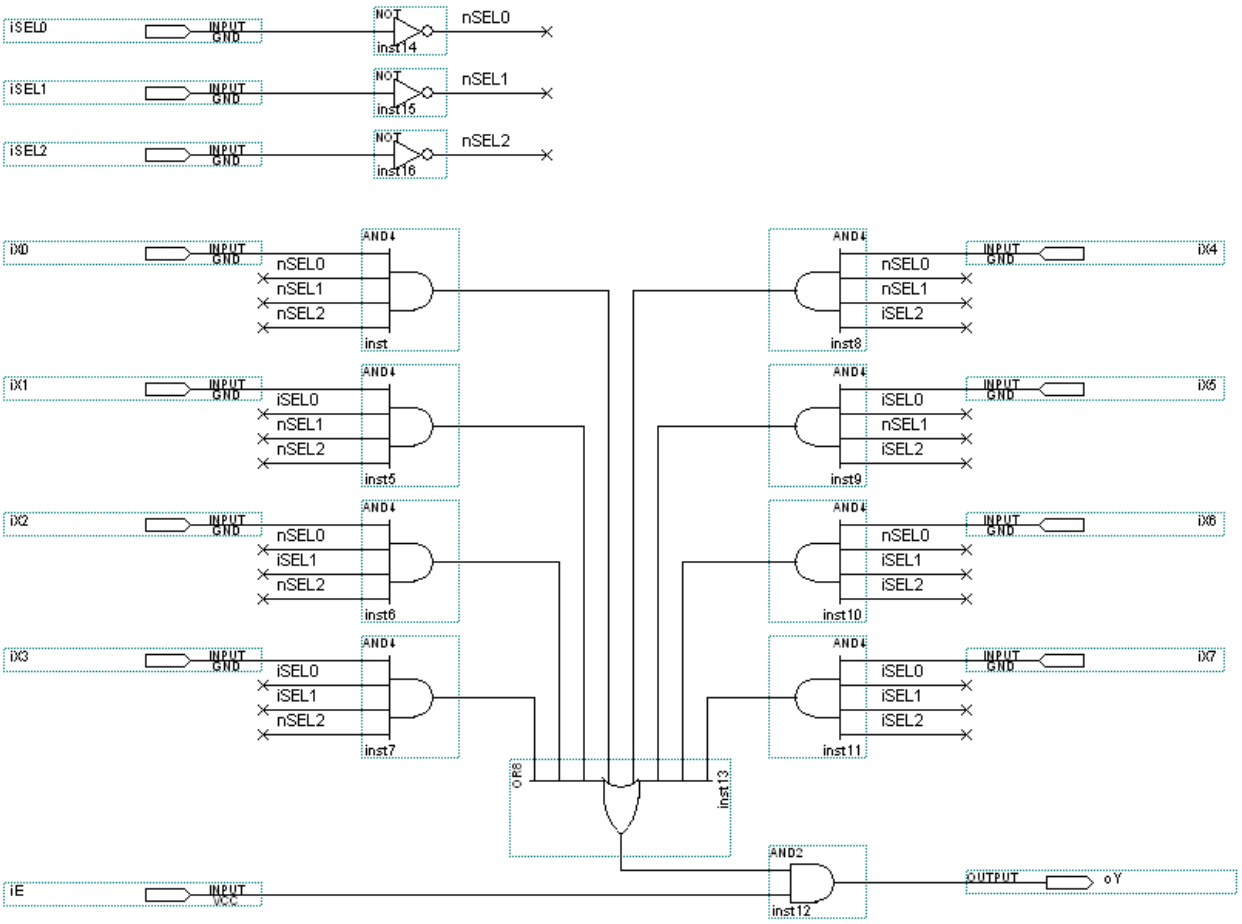
Ulaz označen slovom E predstavlja ulaz dozvole rada multipleksera.

REŠENJE:

Iz tabele se dobija jednačina:

$$Y = \left(\overline{S_2}\cdot\overline{S_1}\cdot\overline{S_0}\cdot X_0 + \overline{S_2}\cdot\overline{S_1}\cdot S_0\cdot X_1 + \overline{S_2}\cdot S_1\cdot\overline{S_0}\cdot X_2 + \overline{S_2}\cdot S_1\cdot S_0\cdot X_3 + S_2\cdot\overline{S_1}\cdot\overline{S_0}\cdot X_4 + S_2\cdot\overline{S_1}\cdot S_0\cdot X_5 + S_2\cdot S_1\cdot\overline{S_0}\cdot X_6 + S_2\cdot S_1\cdot S_0\cdot X_7\right)\cdot E =$$

Izlaz iz multipleksera se kontroliše ulazom dozvole, pa se kao finalna jednačina izlaza dobija izraz $O = I \cdot E$. Na osnovu dobijenih jednačina formira se logička šema multipleksera 8×1 (Slika 3.17).



Slika 3.17: Logička šema multipleksera 8×1

Na osnovu dobijene jednačine se takođe može konstruisati multiplekser u VHDL jeziku za opis fizičke arhitekture:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY MUX8x1 IS PORT(
 iX:  IN  std_logic_vector(7 downto 0); -- vektor ulaznih signala
 iSEL:IN  std_logic_vector(2 downto 0); -- vektor adresnih signala
 iE:  IN  std_logic;                    -- dozvola rada multipleksera
 oY:  OUT std_logic );                  -- izlaz multipleksera
END MUX8x1;
```

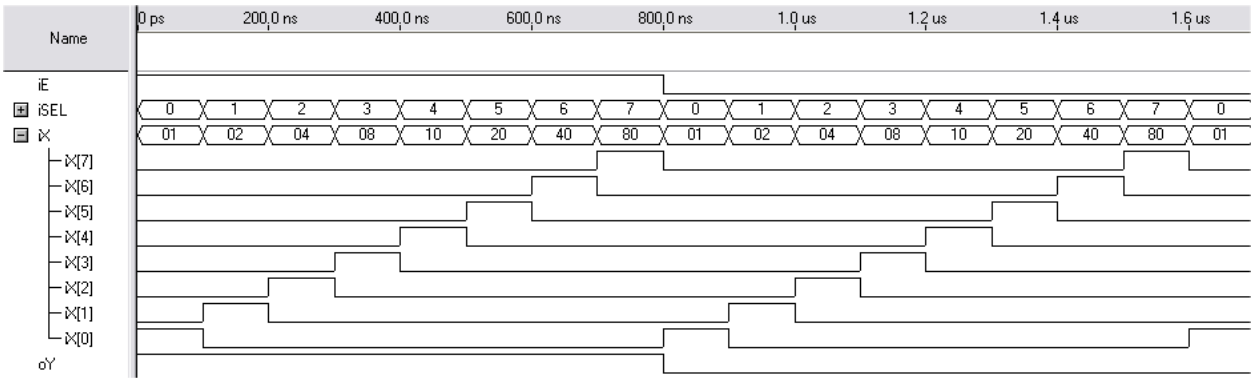
```
ARCHITECTURE ARH_MUX8x1 OF MUX8x1IS BEGIN
  oY <= (iX(0) AND NOT(iSEL(2)) AND NOT(iSEL (1)) AND NOT(iSEL(0))) OR
        (iX(1) AND NOT(iSEL(2)) AND NOT(iSEL (1)) AND    (iSEL(0))) OR
        (iX(2) AND NOT(iSEL(2)) AND    (iSEL (1)) AND NOT(iSEL(0))) OR
        (iX(3) AND NOT(iSEL(2)) AND    (iSEL (1)) AND    (iSEL(0))) OR
        (iX(4) AND    (iSEL(2)) AND NOT(iSEL (1)) AND NOT(iSEL(0))) OR
        (iX(5) AND    (iSEL(2)) AND NOT(iSEL (1)) AND    (iSEL(0))) OR
        (iX(6) AND    (iSEL(2)) AND    (iSEL (1)) AND NOT(iSEL(0))) OR
        (iX(7) AND    (iSEL(2)) AND    (iSEL (1)) AND    (iSEL(0))) OR
END ARH_MUX8x1;
```

Međutim, ovakav pristup projektovanju u VHDL-u je nepraktičan iz razloga što se teško modifikuje kao i zbog nepreglednosti koda. U VHDL-u postoje jezičke konstrukcije pomoću kojih se na elegantan način opisuju multiplekseri. Na primer to su konstrukcije tipa `WITH-SELECT-WHEN` ili `WHEN-ELSE`. Ovde će biti prikazan primer realizacije multipleksera uz pomoć `CASE` konstrukcije. Ovakva konstrukcija zahteva korišćenje procesa za njenu implementaciju, pa multiplekser izgleda ovako:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY MUX8x1 IS PORT (
 iX:   IN  std_logic_vector(7 DOWNTO 0); -- vektor ulaznih signala
 iSEL: IN  std_logic_vector(2 DOWNTO 0); -- vektor adresnih signala
 iE:   IN  std_logic;                    -- dozvola rada multipleksera
 oY:   OUT std_logic );                  -- izlaz multipleksera
END MUX8x1;

ARCHITECTURE ARH_MUX8x1 OF MUX8x1 IS BEGIN
  PROCESS (iX, iSEL, iE) BEGIN
    -- provera vrednosti signala dozvole rada
    IF (iE = '1')THEN
      -- multipleksiranje dozvoljeno ->
      -- odredjivanje izlaznog signala
      -- u zavisnosti od vrednosti adresnog vektora
      CASE iSEL IS
        WHEN "000"  => oY <= iX(0);
        WHEN "001"  => oY <= iX(1);
        WHEN "010"  => oY <= iX(2);
        WHEN "011"  => oY <= iX(3);
        WHEN "100"  => oY <= iX(4);
        WHEN "101"  => oY <= iX(5);
        WHEN "110"  => oY <= iX(6);
        WHEN OTHERS => oY <= iX(7);
      END CASE;
    ELSE
      -- multipleksiranje nije dozvoljeno ->
      -- postavljanje definisane vrednosti na izlaz
      oY <= '0';
    END IF;
  END PROCESS;
END ARH_MUX8x1;
```

Iz samog izvornog koda se uočava da je za proveru vrednosti signala dozvole rada multipleksera (E) iskorišćena `IF-ELSE` konstrukcija. Ovakav kod je funkcionalno potpuno ekvivalentan sa prethodno navedenim kodom, ali se u potpunosti uklapa u filozofiju VHDL programiranja. Prethodno navedeni kod više odgovara filozofiji projektovanja logičkih šema.

Slika 3.18 prikazuje ponašanje u vremenu miltipleksera 8×1. Vidi se da se za odgovarajuću vrednost adresnih promenljivih `iSEL`, na izlazu `oY` pojavljuje vrednost odgovarajućeg ulaznog signala `iX`, uz uslov da je signal dozvole (`iE`) na visokom nivou.



Slika 3.18: Vremenski dijagram multipleksera 8×1

3.5 ZADATAK:

Realizovati demultiplekser sa osam izlaza (Y_7, Y_6, Y_5, Y_4, Y_3, Y_2, Y_1, Y_0), tri adresna ulaza (S_2, S_1, S_0) i jednim ulazom (X) pomoću logičkih kola i pomoću VHDL jezika za opis fizičke arhitekture. Tabela 3.7 prikazuje kombinacionu tabelu traženog demultipleksera, gde je se E označen ulazni signal dozvole rada.

E	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	X
1	0	0	1	0	0	0	0	0	0	X	0
1	0	1	0	0	0	0	0	0	X	0	0
1	0	1	1	0	0	0	0	X	0	0	0
1	1	0	0	0	0	0	X	0	0	0	0
1	1	0	1	0	0	X	0	0	0	0	0
1	1	1	0	0	X	0	0	0	0	0	0
1	1	1	1	X	0	0	0	0	0	0	0

Tabela 3.7: Tabela kojom se opisuje finkcionisanje demultipleksera 1×8

REŠENJE:

Na osnovu tabele se dobijaju izlazne jednačine demultipleksera:

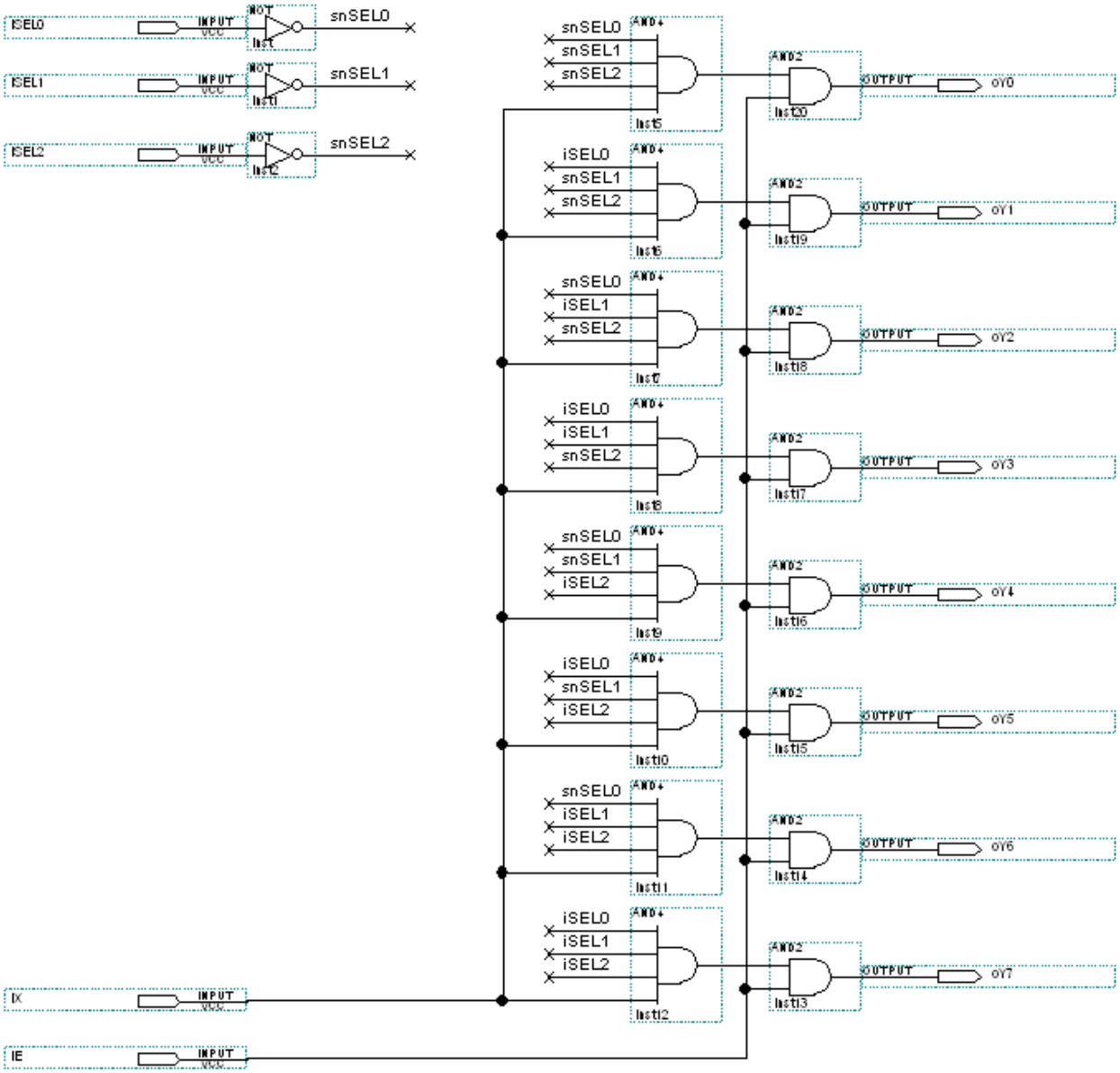
$$Y_0 = \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} \cdot E \cdot X \qquad Y_4 = S_2 \cdot \overline{S_1} \cdot \overline{S_0} \cdot E \cdot X$$

$$Y_1 = \overline{S_2} \cdot \overline{S_1} \cdot S_0 \cdot E \cdot X \qquad Y_5 = S_2 \cdot \overline{S_1} \cdot S_0 \cdot E \cdot X$$

$$Y_2 = \overline{S_2} \cdot S_1 \cdot \overline{S_0} \cdot E \cdot X \qquad Y_6 = S_2 \cdot S_1 \cdot \overline{S_0} \cdot E \cdot X$$

$$Y_3 = \overline{S_2} \cdot S_1 \cdot S_0 \cdot E \cdot X \qquad Y_7 = S_2 \cdot S_1 \cdot S_0 \cdot E \cdot X$$

Na osnovu jednačina demultipleksera može se isprojektovati logička šema koju prikazuje Slika 3.19.



Slika 3.19: Logička šema demultipleksera 1×8

Demultiplekser je takođe moguće isprojektovati u VHDLu realizacijom jednačina izlaza demultipleksera. Međutim, tim pristupom se ne koriste

strukturalne mogućnosti VHDLa, čijim korišćenjem se elegantno opisuju kombinacione mreže proizvoljne složenosti. Jedan primer realizacije demultipleksera u VHDLu je prikazan sledećim izvornim kodom:

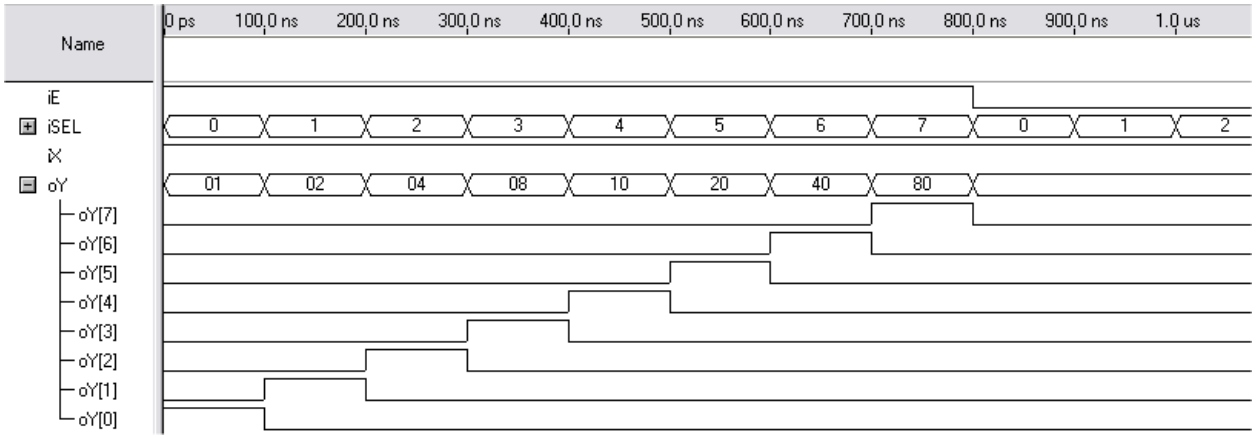
```
LIBRARY IEEE;
  USE ieee.std_logic_1164.all;

ENTITY DMUX1x8 IS PORT(
 iX   : IN  std_logic;                       -- ulazni signal
 iSEL : IN  std_logic_vector (2 DOWNTO 0);   -- vektor adresnih signala
 iE   : IN  std_logic;                       -- signal dozvole rada
 oY   : OUT std_logic_vector (7 DOWNTO 0) );-- vektor izlaznih signala
END DMUX1x8;

ARCHITECTURE ARH_DMUX1x8 OF DMUX1x8 IS BEGIN

  PROCESS (iX, iE, iSEL) BEGIN
    oY <= "00000000"; -- inicijalizacija izlaznog vektora

    -- provera signala dozvole rada
    IF (iE = '1')THEN
      -- demultipleksiranje dozvoljeno ->
      -- prosledjivanje ulaznog signala na izlaz
      -- u zavisnosti od vrednosti adresnog vektora
      CASE iSEL IS
        WHEN "000"  => oY(0) <= iX;
        WHEN "001"  => oY(1) <= iX;
        WHEN "010"  => oY(2) <= iX;
        WHEN "011"  => oY(3) <= iX;
        WHEN "100"  => oY(4) <= iX;
        WHEN "101"  => oY(5) <= iX;
        WHEN "110"  => oY(6) <= iX;
        WHEN OTHERS => oY(7) <= iX;
      END CASE;
    ELSE
      -- demultipleksiranje nije dozvoljeno ->
      -- dodela predefinisane vrednosti izlaznom signalu
      oY <= "00000000";
    END IF;
  END PROCESS;

END ARH_DMUX1x8;
```

Entitet `DMUX1x8` opisuje sprežni sistem demultipleksera. On se sastoji od ulaznog signala `iX` koji se raspoređuje na jedan od izlaznih signala, koji su predstavljeni vektorom izlaznih signala `oY`, u zavisnosti od vrednosti vektora adresnih signala `iSEL`. Ulazni signal `iE` predstavlja signal dozvole rada demultipleksera.

U okviru arhitekture demultiplekser se opisuje jednim procesom. Na početku procesa se vrši postavljanje početnih vrednosti vektora izlaznih signala. Zatim se vrši provera vrednosti signala dozvole `iE`, i ukoliko je on na visokom nivou vrši se demultipleksiranje ulaznog signala na adresirani izlazni signal. U suprotnom se na izlazu forsiraju sve nule.

Detaljnom analizom prikazanog VHDL koda, dolazi se do zaključka da je `ELSE` grana nepotrebna zbog inicijalizacije izlaza na samom početku procesa, pa ako je signal dozvole rada demultipleksera (`iE`) na niskom nivou neće doći do promene vrednosti izlaznog vektora, tj. svi izalazi će biti na niskom nivou.

Slika 3.20 prikazuje vremenske dijagrame rada demultipleksera 1×8.



Slika 3.20: Vremenski dijagram demultipleksera 1×8

3.6 ZADATAK:

a) Realizovati 4-bitni pomerač čije funkcije prikazuje Tabela 3.8 uz pomoć četiri multipleksera 8 na 1 (74151). Tabela 3.9 prikazuje funkcionalnu tabelu multipleksera 74151.
b) Izvršiti sintezu istog pomerača u VHDL jeziku za opis fizičke arhitekture.

S_2	S_1	S_0	Vrsta pomeranja
0	0	0	Propuštanje
0	0	1	shl0
0	1	0	shr0
0	1	1	shlc
1	0	0	shrc
1	0	1	ashl
1	1	0	ashr
1	1	1	ashlc

Tabela 3.8: Funkcije pomerača

U slučaju aritmetičkog pomeranja, ulazne slogove posmatrati u II komplementu.

INPUTS				OUTPUTS	
SELECT			STROBE	Y	W
C	B	A	$\overline{G}$		
×	×	×	H	L	H
L	L	L	L	D_0	$\overline{D_0}$
L	L	H	L	D_1	$\overline{D_1}$
L	H	L	L	D_2	$\overline{D_2}$
L	H	H	L	D_3	$\overline{D_3}$
H	L	L	L	D_4	$\overline{D_4}$
H	L	H	L	D_5	$\overline{D_5}$
H	H	L	L	D_6	$\overline{D_6}$
H	H	H	L	D_7	$\overline{D_7}$

Tabela 3.9: Funkcionalna tabela multipleksera 74LS151

REŠENJE:

a)

Na osnovu operacija koje treba realizovati mogu se formirati izlazi pomerača, za svaku operaciju posebno. To prikazuje Tabela 3.10.

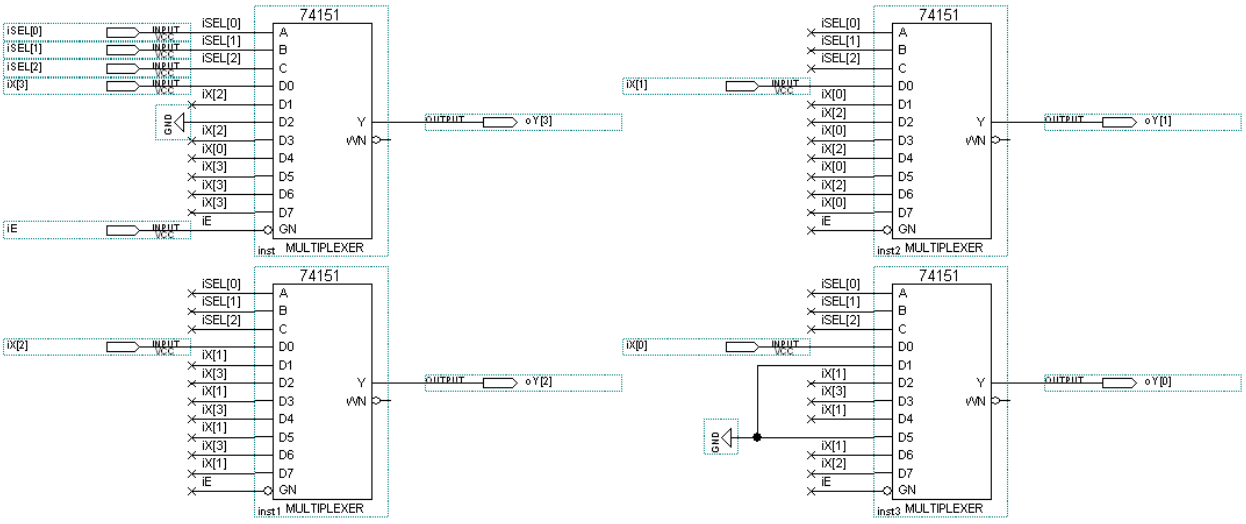
S_2	S_1	S_0	Y_3	Y_2	Y_1	Y_0	Vrsta pomeranja
0	0	0	X_3	X_2	X_1	X_0	Propuštanje
0	0	1	X_2	X_1	X_0	0	shl0
0	1	0	0	X_3	X_2	X_1	shr0
0	1	1	X_2	X_1	X_0	X_3	shlc
1	0	0	X_0	X_3	X_2	X_1	shrc
1	0	1	X_3	X_1	X_0	0	ashl
1	1	0	X_3	X_3	X_2	X_1	ashr
1	1	1	X_3	X_1	X_0	X_2	ashlc

Tabela 3.10: Tabela operacija pomeranja koje realizuje pomerač

Iz prethodne tabele jednostavno se vrši iščitavanje signala koji predstavljaju ulaze multipleksera u zavisnosti od selekcionog ulaza. Tako na primer, u slučaju izlaza pomerača Y_3, ako je selekcioni ulaz multipleksera "000", na njegov D_0 ulaz dovodi se signal X_3. Ako je selekcioni ulaz "001" na njegov D_1 ulaz dovodi se signal X_2, i tako redom, sve do selekcionog ulaza "111" kada se na D_7 ulaz multipleksera dovodi signal '0' (GND). Analogan postupak se vrši i za preostale izlaze pomerača Y_0, Y_1 i Y_2.

Adresni ulazi pomerača S_2, S_1 i S_0 se dovode direktno na adresne ulaze svih multipleksera C, B i A redom.

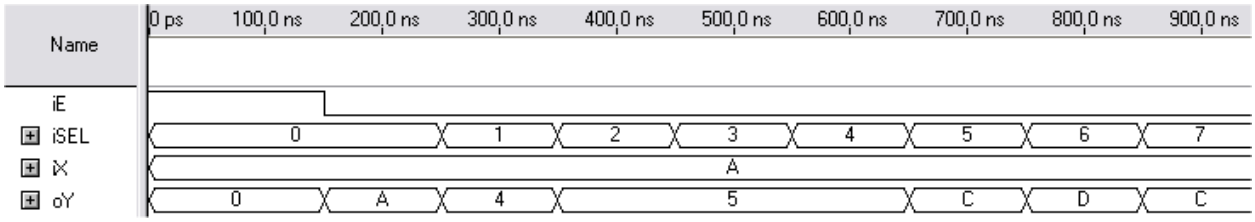
Slika 3.21 prikazuje realizaciju traženog pomerača pomoću multipleksera.



Slika 3.21: Logička šema traženog pomerača

Na logičku šemu je pored ulaznog vektora i adresnih signala uveden i ulazni signal dozvole pomeranja `iE`, koji se dovodi na `STROBE` (`GN` na šemi) ulaz multipleksera. Ako je vrednost ovog signala jednaka nuli, tada se izvršava pomeranje u zavisnosti od adresnih ulaza pomerača. Inače, na izlazu pomerača se forsira vrednost nula.

Simulaciju rada realizovanog pomerača prikazuje Slika 3.22. Vremenski dijagram prikazuje sve vrste pomeranja ulaznog vektora sa vrednošću A heksadecimalno.



Slika 3.22: Simulacija rada pomerača

b)

VHDL sinteza traženog pomerača sledi direktno iz tabele operacija koje izvršava pomerač, Tabela 3.10. Kao i na logičkoj šemi i ovde treba realizovati tri ulazna signala (ulazni vektor `iX`, adresni vektor `iSEL` i signal dozvole pomeranja `iE`) i jedan izlaz koji predstavlja rezultat pomeranja, `oY`.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY POMERAC IS PORT (
  iX:   IN  std_logic_vector(3 DOWNTO 0);
  iSEL: IN  std_logic_vector(2 DOWNTO 0);
  iE:   IN  std_logic;
  oY:   OUT std_logic_vector(3 DOWNTO 0));
END POMERAC;

ARCHITECTURE ARH_POMERAC OF POMERAC IS
BEGIN

  -- proces koji opisuje kombinacionu mrezu pomeraca
  -- koja na osnovu adresnog ulaza i signala dozvole pomeranja
  -- izvrsava zadatu vrstu pomeranja
  PROCESS (iX, iSEL, iE) BEGIN
    IF (iE = '0') THEN
      CASE iSEL IS
            -- propustanje
        WHEN "000"  => oY <= iX;
            -- sh10
        WHEN "001"  => oY <= (iX(2 DOWNTO 0) & '0');
            -- shr0
        WHEN "010"  => oY <= ('0' & iX(3 DOWNTO 1));
            -- sh1c
        WHEN "011"  => oY <= (iX(2 DOWNTO 0) & iX(3));
            -- shrc
        WHEN "100"  => oY <= (iX(0) & iX(3 DOWNTO 1));
            -- ash1
```

```
            WHEN "101"  => oY <= (iX(3) & iX(1) & iX(0) & '0');
                 -- ashr0
            WHEN "110"  => oY <= (iX(3) & iX(3) & iX(2) & iX(1));
                 -- ashlc
            WHEN "111"  => oY <= (iX(3) & iX(1) & iX(0) & iX(2));
                 -- inace pomeranje nije dozvoljeno
            WHEN OTHERS => oY <= "0000";
          END CASE;
        ELSE
          oY <= "0000"; -- pomeranje nije dozvoljeno
        END IF;
      END PROCESS;

    END ARH_POMERAC;
```

U arhitekturi realizovanog pomerača ARH_POMERAC, postoji samo jedan proces koji opisuje traženi pomerač. U tom procesu se prvo proverava vrednost signala dozvole pomeranja iE. Ako je pomeranje dozvoljeno, iE=0, tada se analizira adresni vektor i na osnovu njegove vrednosti se izvršava adresirana vrsta pomeranja. Pomeranje je u ovom slučaju realizovano odgovarajućom agregacijom signala ulaznog vektora. U slučaju da pomeranje nije dozvoljeno, iE=1, na izlazu se forsira vektor sa vrednošću nula.

Odziv realizovanog VHDL koda je isti kao i kod realizovane logičke šeme, Slika 3.22.

3.7 ZADATAK:

Pomoću standardnih logičkih kola (I, ILI, NE) izvršiti sintezu 4-bitnog pomerača čije funkcije prikazuje Tabela 3.11.

Pomerač treba da ima četvorobitni ulazni vektor X i dvobitni adresni vektor S. Na izlazu treba da obezbedi rezultat pomeranju u obliku četvorobitnog vektora Y.

S_1	S_0	Operacija
0	0	ZERO
0	1	shl X
1	0	shr X
1	1	propuštanje

Tabela 3.11: Funkcije pomerača

REŠENJE:

Tabela 3.12 prikazuje prenosnu funkciju pomerača.

Operacija	S_1	S_0	Y_3	Y_2	Y_1	Y_0
ZERO	0	0	0	0	0	0
shl X	0	1	X_2	X_1	X_0	0
shr X	1	0	0	X_3	X_2	X_1
propuštanje	1	1	X_3	X_2	X_1	X_0

Tabela 3.12: Prenosna funkcija pomerača

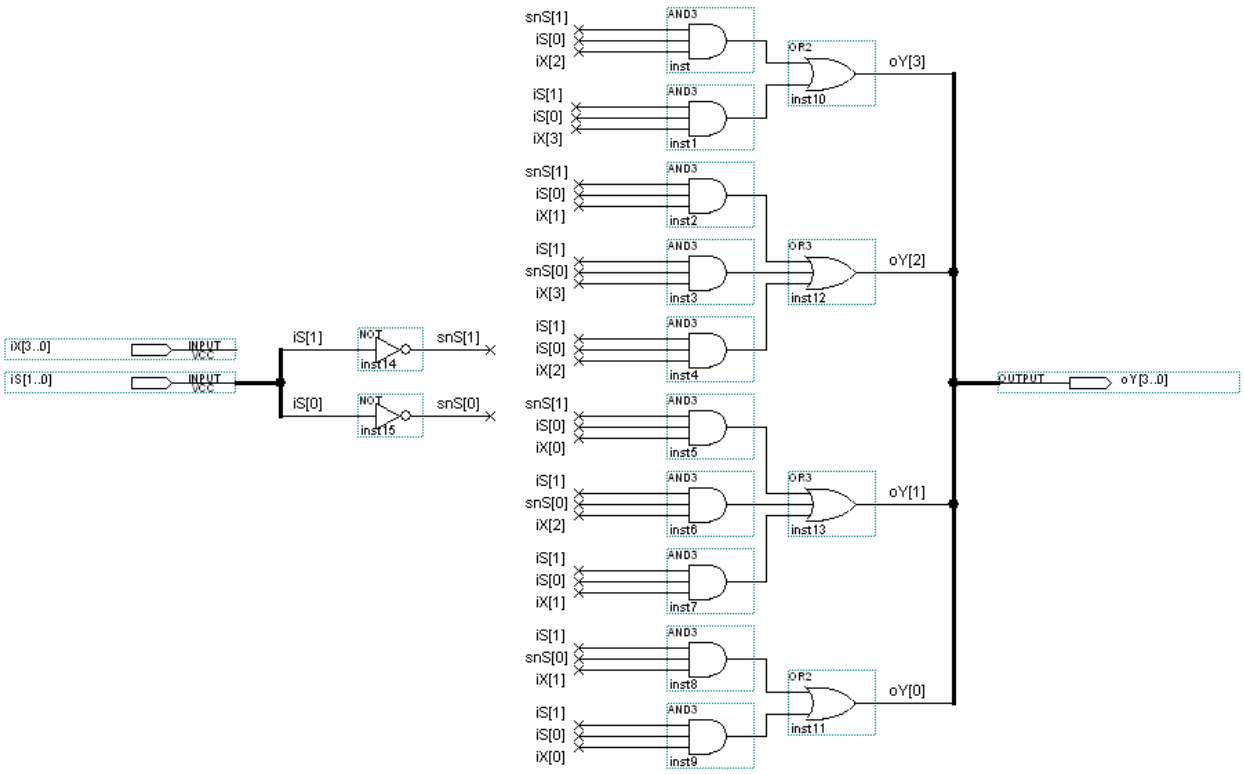
Na osnovu prethodne tabele formiraju se prenosne funkcije sva četiri izlaza (Y_3, Y_2, Y_1 i Y_0) koje glase:

$$Y_3 = 0 \cdot \overline{S_1} \cdot \overline{S_0} + X_2 \cdot \overline{S_1} \cdot S_0 + 0 \cdot S_1 \cdot \overline{S_0} + X_3 \cdot S_1 \cdot S_0$$
$$Y_2 = 0 \cdot \overline{S_1} \cdot \overline{S_0} + X_1 \cdot \overline{S_1} \cdot S_0 + X_3 \cdot S_1 \cdot \overline{S_0} + X_2 \cdot S_1 \cdot S_0$$
$$Y_1 = 0 \cdot \overline{S_1} \cdot \overline{S_0} + X_0 \cdot \overline{S_1} \cdot S_0 + X_2 \cdot S_1 \cdot \overline{S_0} + X_1 \cdot S_1 \cdot S_0$$
$$Y_0 = 0 \cdot \overline{S_1} \cdot \overline{S_0} + 0 \cdot \overline{S_1} \cdot S_0 + X_1 \cdot S_1 \cdot \overline{S_0} + X_0 \cdot S_1 \cdot S_0$$

Ukoliko se eleminišu članovi uz koje figuriše nula, dobija se da u prenosnim funkcijama pomerača Y_3 i Y_0 figurišu samo dva prizvoda, a u funkcijama Y_2 i Y_1 po tri proizvoda:

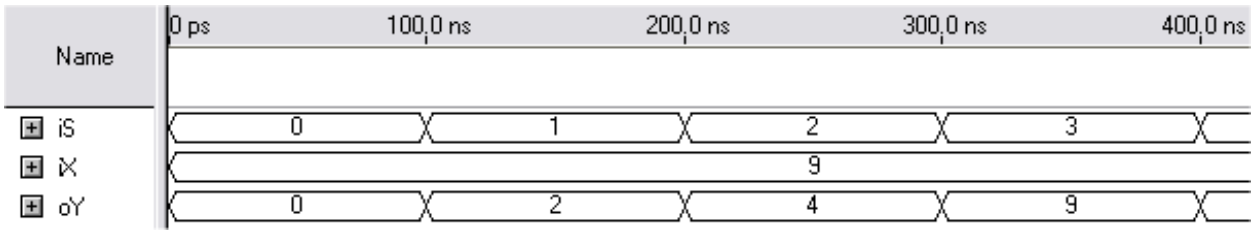
$$Y_3 = X_2 \cdot \overline{S_1} \cdot S_0 + X_3 \cdot S_1 \cdot S_0$$
$$Y_2 = X_1 \cdot \overline{S_1} \cdot S_0 + X_3 \cdot S_1 \cdot \overline{S_0} + X_2 \cdot S_1 \cdot S_0$$
$$Y_1 = X_0 \cdot \overline{S_1} \cdot S_0 + X_2 \cdot S_1 \cdot \overline{S_0} + X_1 \cdot S_1 \cdot S_0$$
$$Y_0 = X_1 \cdot S_1 \cdot \overline{S_0} + X_0 \cdot S_1 \cdot S_0$$

Time su formirane prenosne funkcije na osnovu kojih se pristupa sintezi logičke šeme pomerača, Slika 3.23.



Slika 3.23: Logička šema realizovanog pomerača

Simulaciju rada realizovanog pomerača prikazuje Slika 3.24. Vremenski dijagram prikazuje sve vrste pomeranja ulaznog vektora sa vrednošću 9 heksadecimalno.



Slika 3.24: Simulacija rada pomerača

3.8 ZADATAK:

Izvršiti sintezu dvosmernog četvorobitnog pomerača pomoću VHDL jezika za opis fizičke arhitekture. Tabela 3.13 prikazuje smer i vrstu pomeranja u zavisnosti od adresnog ulaza S. Sva pomeranja realizovati kao logička pomeranja sa ubacivanjem nula.

	Smer	Broj bita		Vrsta pomeranja
	S_2	S_1	S_0	
ulevo	0	0	0	propuštanje
	0	0	1	shl 1 mesto
	0	1	0	shl 2 mesta
	0	1	1	shl 3 mesta
udesno	1	0	0	propuštanje
	1	0	1	shr 1 mesto
	1	1	0	shr 2 mesta
	1	1	1	shr 3 mesta

Tabela 3.13: Funkcije pomerača

REŠENJE:

Radi jasnijeg pregleda operacija koje treba izvršiti u zavisnosti od vrednosti adresnih ulaza, formira se funkcionalna tabela pomerača koja prikazuje vrednost izlaznog vektora Y u zavisnosti od ulaznog vektora X i vrednosti adresnih ulaza S, Tabela 3.14.

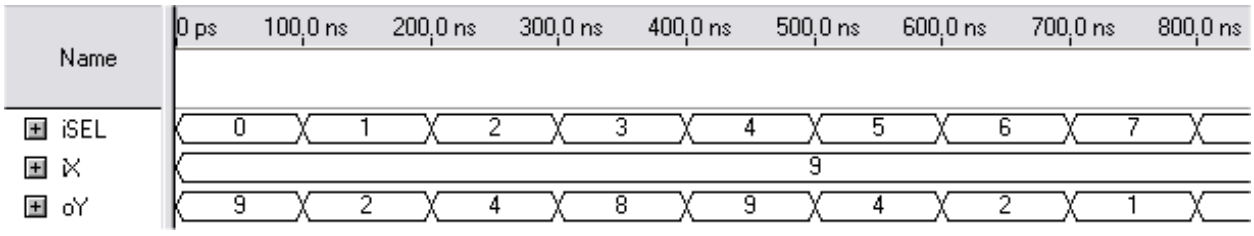
	Smer	Broj bita		Vrednost izlaza				Vrsta pomeranja
	S_2	S_1	S_0	Y_3	Y_2	Y_1	Y_0	
ulevo	0	0	0	X_3	X_2	X_1	X_0	propuštanje
	0	0	1	X_2	X_1	X_0	0	shl 1 mesto
	0	1	0	X_1	X_0	0	0	shl 2 mesta
	0	1	1	X_0	0	0	0	shl 3 mesta
udesno	1	0	0	X_3	X_2	X_1	X_0	propuštanje
	1	0	1	0	X_3	X_2	X_1	shr 1 mesto
	1	1	0	0	0	X_3	X_2	shr 2 mesta
	1	1	1	0	0	0	X_3	shr 3 mesta

Tabela 3.14: Funkcionalana tabela traženog pomerača

Na osnovu prethodne tabele sledi implementacija VHDL opisa pomerača. Traženi pomerač se može opisati pomoću jednog procesa koji opisuje kombinacionu mrežu koja na osnovu vrednosti adresnih ulaza (CASE iskaz) realizuje potrebno pomeranje.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY POMERAC IS PORT (
  iX:   IN  std_logic_vector(3 DOWNTO 0);
  iSEL: IN  std_logic_vector(2 DOWNTO 0);
  oY:   OUT std_logic_vector(3 DOWNTO 0));
END POMERAC;

ARCHITECTURE ARH_POMERAC OF POMERAC IS
BEGIN

  -- proces koji opisuje kombinacionu mrezu pomeraca
  -- koja na osnovu adresnog ulaza
  -- izvrsava zadatu vrstu pomeranja
  PROCESS (iX, iSEL) BEGIN
    CASE iSEL IS
      WHEN "000"  => oY <= iX;                       -- propustanje
      WHEN "001"  => oY <= (iX(2 DOWNTO 0) & '0');   -- shl 1 mesto
      WHEN "010"  => oY <= (iX(1 DOWNTO 0) & "00");  -- shl 2 mesta
      WHEN "011"  => oY <= (iX(0) & "000");          -- shl 2 mesta
      WHEN "100"  => oY <= iX;                       -- propustanje
      WHEN "101"  => oY <= ('0' & iX(3 DOWNTO 1));   -- shr 1 mesto
      WHEN "110"  => oY <= ("00" & iX(3 DOWNTO 2));  -- shr 2 mesta
      WHEN OTHERS => oY <= ("000" & iX(3));          -- shr 3 mesta
    END CASE;
  END PROCESS;

END ARH_POMERAC;
```

Simulaciju rada realizovanog pomerača prikazuje Slika 3.25. Vremenski dijagram prikazuje sve vrste pomeranja ulaznog vektora sa vrednošću 9 heksadecimalno.



Slika 3.25: Simulacija rada pomerača

3.9 ZADATAK:

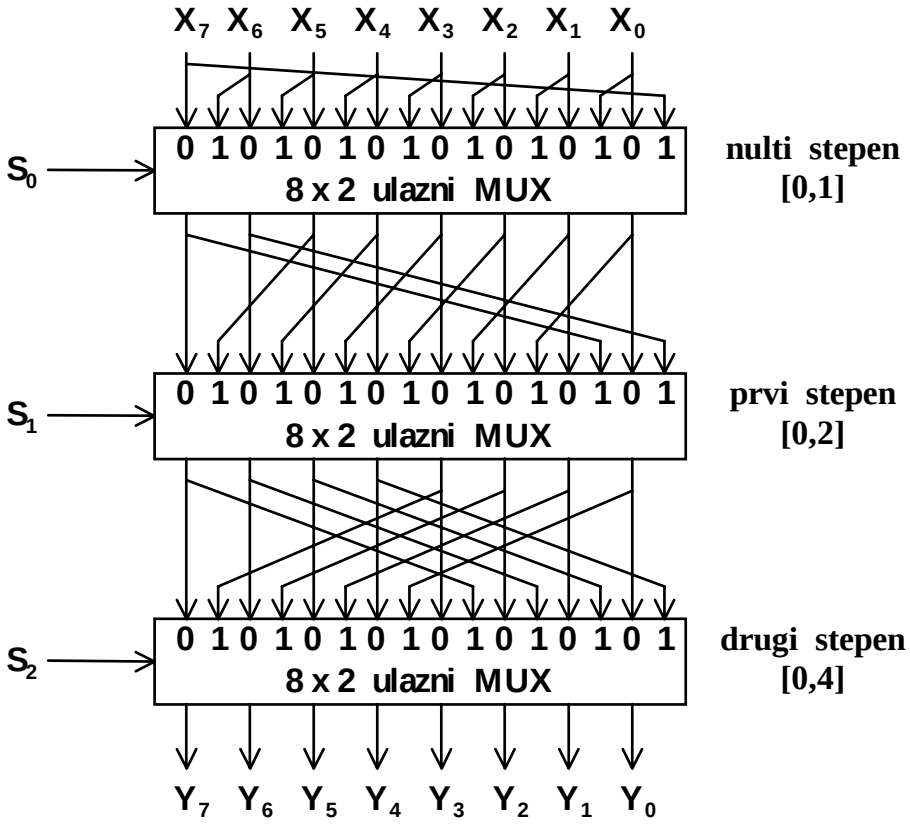
Pomoću VHDL jezika za opis fizičke arhitekture realizovati trostepeni Barelov pomerač osmobitnih vrednosti.

REŠENJE:

Barelov pomerač je mreža od r stepeni takva da i-ti stepen pomera na rastojanje 0 ili 2^i. Svaki stepen se realizuje kao n × 2 ulazni multiplekser gde se adresiranje ulaza i-tog multipleksera određuje sa bitom S_i.

Svaki stepen trostepenog barelovog pomerača se realizuje u obliku n × 2 ulaznog multipleksera kao što prikazuje Slika 3.26.

Slika 3.26 prikazuje da u zavisnosti od vrednosti ulazne promenljive S_0 nulti stepen ili propušta ulaznu reč X (slučaj kada je S_0=0) ili je pomera za jedno (1=2^0) mesto u levo (slučaj kada je S_0=1). Slično nultom stepenu, prvi stepen u zavisnosti od vrednosti ulazne promenljive S_1 ili propušta izlaz iz nultog stepena (slučaj kada je S_1=0) ili ga pomera za dva (2=2^1) mesta u levo (slučaj kada je S_1=1). Analogno, drugi stepen u zavisnosti od ulazne promenljive S_2 ili propušta izlaz iz prvog stepena ili ga pomera za četiri (4=2^2) mesta u levo.

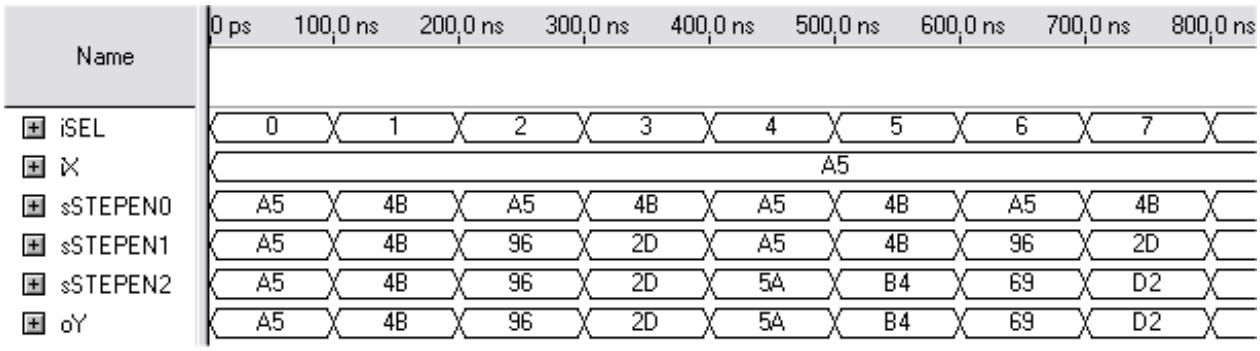


Slika 3.26: Šema osmobitnog trostepenog Barelovog pomerača

Na osnovu prethodne analize uočava se da traženi pomerač za ulaze ima vektor osmobitnih ulaznih vrednosti i trobitni vektor adresnih signala za odabir stepeni koji učestvuju u pomeranju. Od izlaza postoji samo osmobitni vektor izlaznih vrednosti koji predstavlja izlaz pomerača. VHDL kod realizacije takvog pomerača je prikazan u nastavku:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY BARREL IS PORT (
 -- vektor ulaznih  signala
 iX:   IN  std_logic_vector(7 DOWNTO 0);
 -- vektor adresnih signala
 iSEL: IN  std_logic_vector(2 DOWNTO 0);
 -- vektor izlaznih signala
 oY:   OUT std_logic_vector(7 DOWNTO 0)
);
END BARREL;

ARCHITECTURE ARH_BARREL OF BARREL IS
-- vektori koji predstavljaju izlaze
-- odgovarajucih stepeni Barelovog pomeraca
SIGNAL sSTEPEN0, sSTEPEN1, sSTEPEN2: std_logic_vector(7 DOWNTO 0);
BEGIN

 -- stepen 0:
 -- pomera ulazni vektor za jedno mesto u levo
 -- ako je adresni signal iSEL(0) jednak jednici
 -- u suprotnom nema pomeranja, propusta se ulazni vektor
 sSTEPEN0 <= (iX(6 DOWNTO 0) & iX(7))
             WHEN (iSEL(0) = '1')
             ELSE iX;

 -- stepen 1:
 -- pomera izlaz nultog stepena za dva mesta u levo
 -- ako je adresni signal iSEL(1) jednak jednici
 -- u suprotnom nema pomeranja, propusta se izlaz nultog stepena
 sSTEPEN1 <= (sSTEPEN0(5 DOWNTO 0) & sSTEPEN0(7 DOWNTO 6))
             WHEN (iSEL(1) = '1')
             ELSE sSTEPEN0;
 -- stepen 2:
 -- pomera izlaz prvog stepena za cetiri mesta u levo
 -- ako je adresni signal iSEL(2) jednak jednici
 -- u suprotnom nema pomeranja, propusta se izlaz prvog stepena
 sSTEPEN2 <= (sSTEPEN1(3 DOWNTO 0) & sSTEPEN1(7 DOWNTO 4))
             WHEN (iSEL(2) = '1')
             ELSE sSTEPEN1;

 -- izlaz trostepenog Barelovog pomeraca je izlaz drugog stepena
 oY <= sSTEPEN2;

END ARH_BARREL;
```

Sa vremenskog dijagrama (Slika 3.27) koji ilustruje rad realizovanog pomerača se vidi da svaki stepen Barelovog pomerača pomera samo ako je odgovarajući adresni ulaz aktivan, tj. nalazi se na visokom potencijalu. Takođe, vidi se da izlaz pomerača predstavlja ulazni vektor pomeren za onoliko mesta koliko predstavlja vrednost adresnog ulaznog vektora.



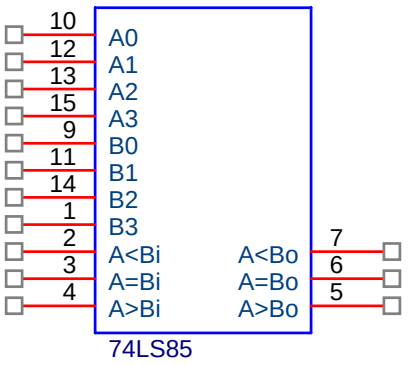
Slika 3.27: Vremenski dijagram rada realizovanog pomerača

3.10 ZADATAK:

Realizovati četvorobitni komparator funkcionalno ekvivalentan sa TTL komponentom sa oznakom 74LS85 koju prikazuje Slika 3.28.

Tabela 3.15 prikazuje funkconisanje komparatora 74LS85.

Komparator realizovati uz pomoć logičkih kola i VHDL jezika za opis fizičke arhitekture.



Slika 3.28: Logički simbol komparatora 74LS85

A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	H_3' A > B	H_2' A < B	H_1' A = B	H_3 A > B	H_2 A < B	H_1 A = B
$A_3 > B_3$	×	×	×	×	×	×	H	L	L
$A_3 < B_3$	×	×	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 > B_2$	×	×	×	×	×	H	L	L
$A_3 = B_3$	$A_2 < B_2$	×	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	×	×	×	×	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	×	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	×	×	×	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	×	×	×	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	H	L	L	H

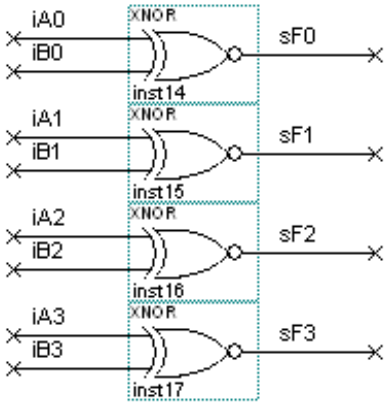
Tabela 3.15: Funkcionisanje komparatora 74LS85

REŠENJE:

Prvo je potrebno realizovati kolo koje proverava jednakost dva nezavisna bita. Relacija jednakosti individualnih bitova može se izraziti sledećom Bulovom funkcijom:

$$F_i = A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}; \qquad i=0, 1, 2, 3$$

gde je $F_i = 1$, ako je par bitova na poziciji (i) jednak (oba su jednaka 1 ili su oba jedanka 0). Slika 3.29 prikazuje realizaciju funkcija F_i uz pomoć dvoulaznih I i ILI logičkih kola.



Slika 3.29: Realizacija funkcija F_i

Jednakost brojeva A i B data je binarnom promenjlivom koju označavamo simbolom (A=B). Ova promenljiva ima vrednost 1 ako je A aritmetički jednako B, a 0 u svim drugim slučajevima. Da bi binarna promenljiva (A=B) bila jednaka 1, sve promenljive F_i moraju biti 1 što diktira funkciju:

$$(A = B) = F_3 \cdot F_2 \cdot F_1 \cdot F_0;$$

Da bi se odredilo da li je A veće ili manje od B, ispituje se relativan moduo parova značajnih bitova počev od bita najveće težine. Ako su ta dva bita jednaka, porede se sledeća dva bita manje težine i nastavlja se sa ovakvim poređenjem dok se ne pronađe par nejednakih bitova. Ako je odgovarajući bit A veći od bita B, tada je A>B, a ako je bit B veći od bita A, tada je B>A.

Ako se označi sa H_1 izlaz A=B, sa H_2 izlaz A<B i sa H_3 izlaz A>B dobijaju se sledeće izlazne jednačine:

$$H_1 = F_3 \cdot F_2 \cdot F_1 \cdot F_0$$

$$H_2 = \overline{A_3} \cdot B_3 + F_3 \cdot \overline{A_2} \cdot B_2 + F_3 \cdot F_2 \cdot \overline{A_1} \cdot B_1 + F_3 \cdot F_2 \cdot F_1 \cdot \overline{A_0} \cdot B_0 + F_3 \cdot F_2 \cdot F_1 \cdot F_0$$

$$H_3 = A_3 \cdot \overline{B_3} + F_3 \cdot A_2 \cdot \overline{B_2} + F_3 \cdot F_2 \cdot A_1 \cdot \overline{B_1} + F_3 \cdot F_2 \cdot F_1 \cdot A_0 \cdot \overline{B_0} + F_3 \cdot F_2 \cdot F_1 \cdot F_0$$

Ulazi H_1', H_2' i H_3' služe za povezivanje komparatora radi upoređivanja vrednosti. Pri tome ulaz H_1' označava da su 4 bitna reči prethodnog stepena jednake, H_2' označava da je reč A prethodnog stepena manja od reči B prethodnog stepena dok H_3' označava da je reč A prethodnog stepena veća od reči B prethodnog stepena.

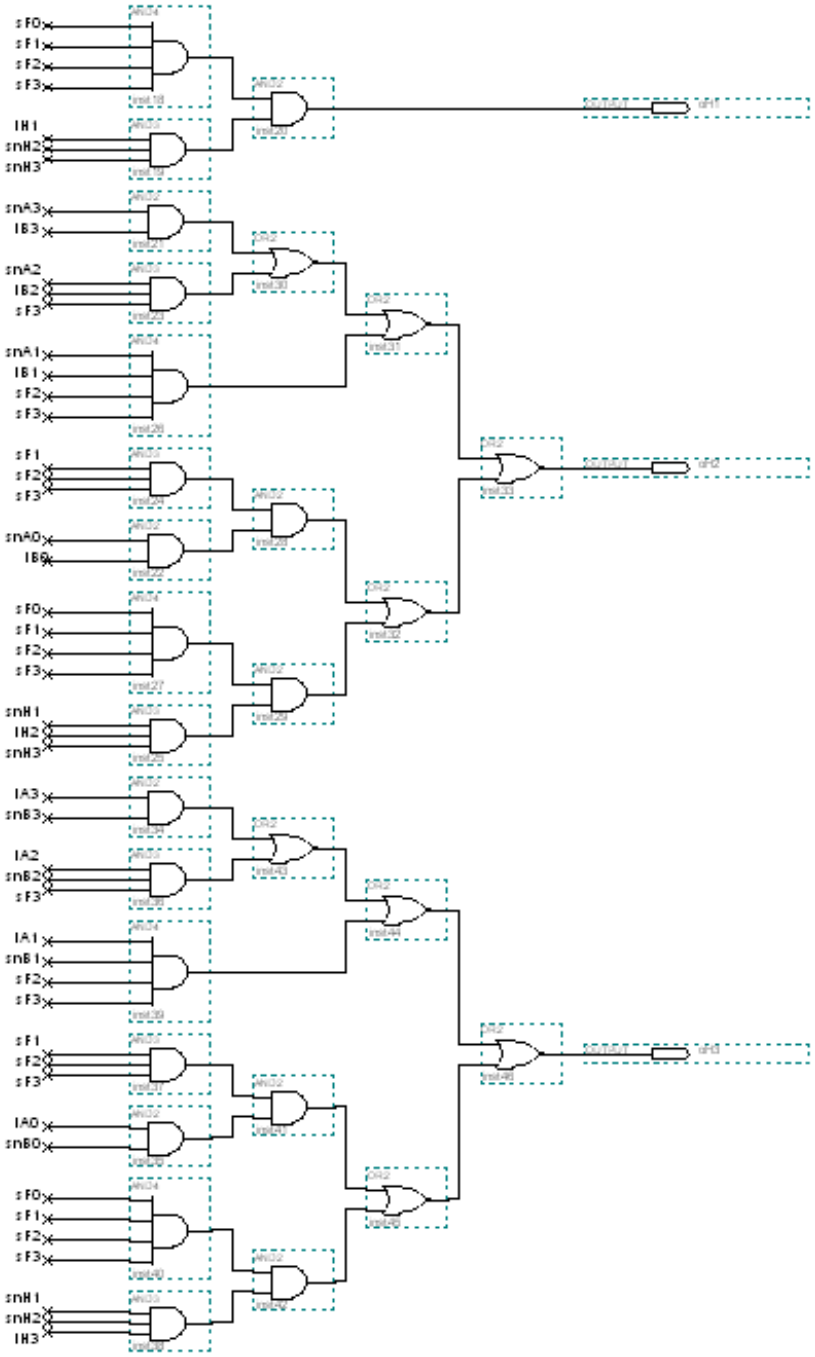
Na osnovu tabele mogu se modifikovati jednačine izlaza komparatora tako da se uključe kaskadni ulazi H_1', H_2' i H_3'.

$$H_1 = F_3 \cdot F_2 \cdot F_1 \cdot F_0 \cdot \overline{H_3'} \cdot \overline{H_2'} \cdot H_1'$$

$$H_2 = \overline{A_3} \cdot B_3 + F_3 \cdot \overline{A_2} \cdot B_2 + F_3 \cdot F_2 \cdot \overline{A_1} \cdot B_1 + F_3 \cdot F_2 \cdot F_1 \cdot \overline{A_0} \cdot B_0 +$$
$$F_3 \cdot F_2 \cdot F_1 \cdot F_0 \cdot \overline{H_3'} \cdot H_2' \cdot \overline{H_1'}$$

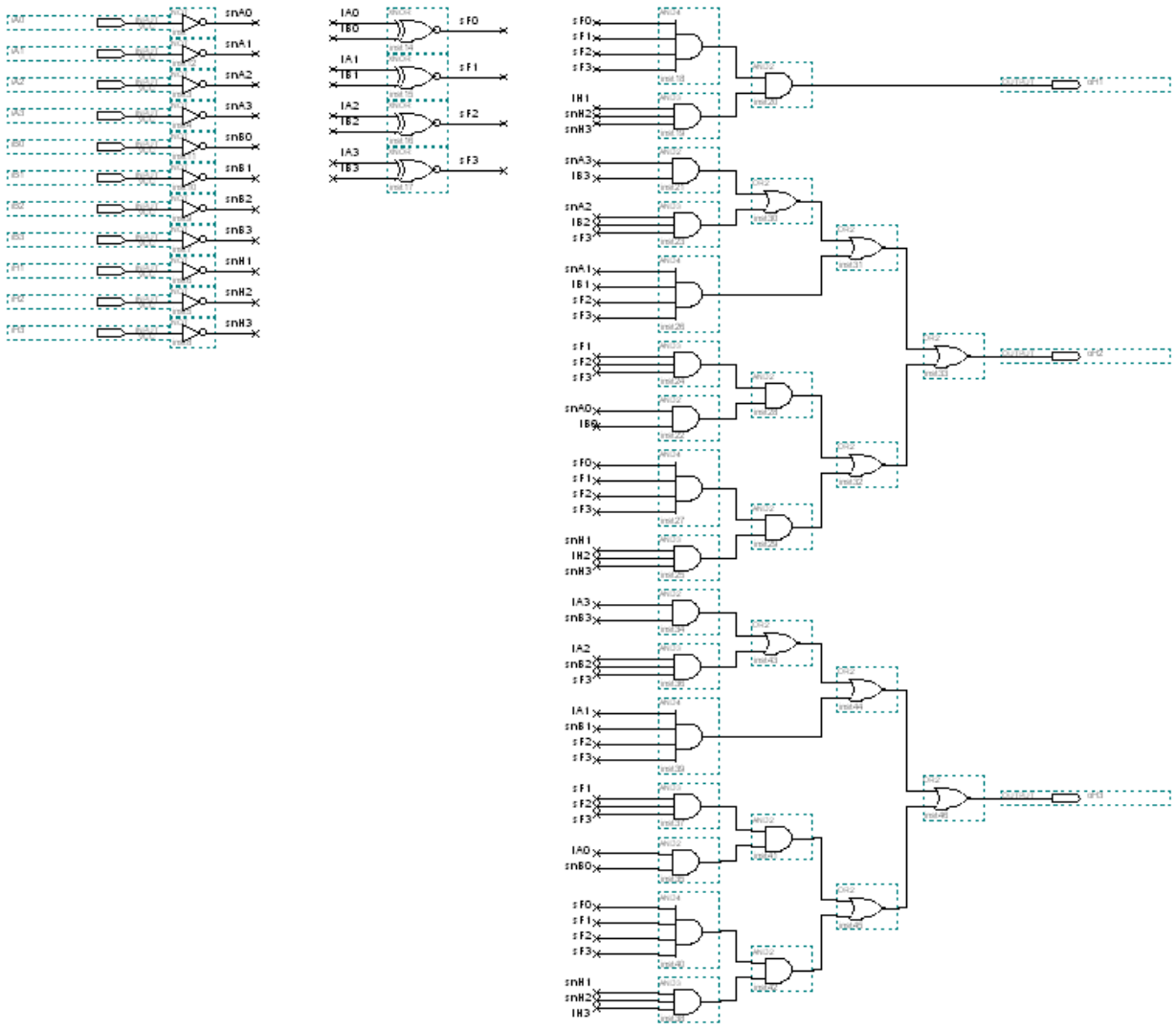
$$H_3 = A_3 \cdot \overline{B_3} + F_3 \cdot A_2 \cdot \overline{B_2} + F_3 \cdot F_2 \cdot A_1 \cdot \overline{B_1} + F_3 \cdot F_2 \cdot F_1 \cdot A_0 \cdot \overline{B_0} +$$
$$F_3 \cdot F_2 \cdot F_1 \cdot F_0 \cdot H_3' \cdot \overline{H_2'} \cdot \overline{H_1'}$$

Realizaciju izlaza komparatora 74LS85 na osnovu ulaznih četvorobitnih vrednosti, kaskadnih ulaza i realizovanih funkcija F_i prikazuje Slika 3.30.



Slika 3.30: Realizacija funkcija H_i

Slika 3.31 prikazuje kompletnu logičku šemu realizacije komparatora 74LS85.



Slika 3.31: Logička šema komparatora 74LS85

Ekvivalentan komparator je moguće realizovati i u VHDL jeziku za opis fizičke arhitekture. Jedna realizacija komparatora je prikazana u sledećem izvornom kodu:

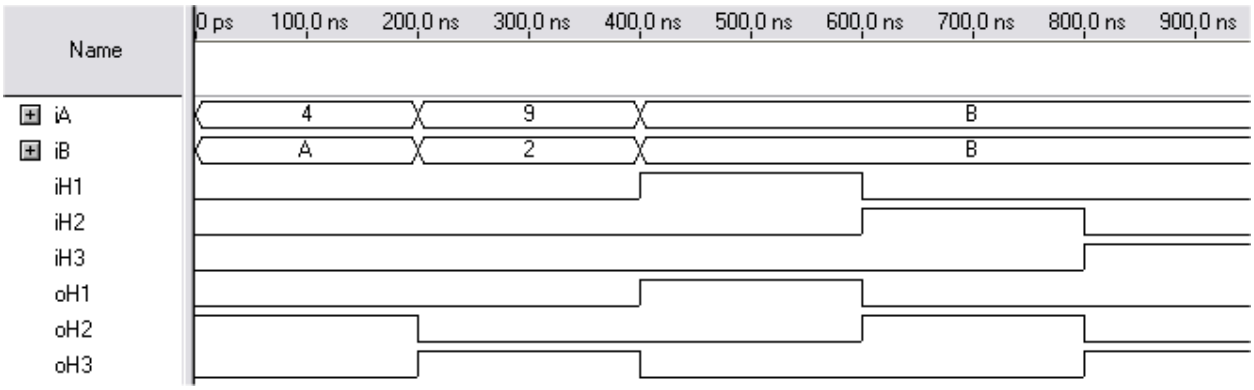
```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY KOMPARATOR IS PORT (
  -- ulazni brojevi
  iA, iB:        IN  std_logic_vector(3 DOWNTO 0);
  -- kaskadni ulazi
  iH3, iH2, iH1: IN  std_logic;
  -- izlazi komparatora
  oH3, oH2, oH1: OUT std_logic );
    -- H1 -> označava da je A=B
    -- H2 -> označava da je A<B
    -- H3 -> označava da je A>B
END KOMPARATOR;
```

```
ARCHITECTURE ARH_KOMPARATOR OF KOMPARATOR IS
BEGIN

  PROCESS (iA, iB, iH1, iH2, iH3) BEGIN
    IF    (iA(3) > iB(3)) THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iA(3) < iB(3)) THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSIF (iA(2) > iB(2)) THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iA(2) < iB(2)) THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSIF (iA(1) > iB(1)) THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iA(1) < iB(1)) THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSIF (iA(0) > iB(0)) THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iA(0) < iB(0)) THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSIF (iH3 = '1')     THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iH2 = '1')     THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSE                       oH3 <= '0'; oH2 <= '0'; oH1 <= '1';
    END IF;
  END PROCESS;

END ARH_KOMPARATOR;
```

Entitet `KOMPARATOR` sadrži ulazno/izlazne signale komparatora. Sa `iA` i `iB` su predstavljeni ulazni brojevi koji se porede. `iH1`, `iH2` i `iH3` su kaskadni ulazi komparatora koji označavaju da je na prethodnom stepenu A=B, A<B i A>B respektivno. Izlazi komparatora su predstavljeni sa signalima `oH1`, `oH2` i `oH3`, koji su aktivni (imaju vrednost 1) ako su ulazni brojevi redom u sledećim relacijama A=B, A<B i A>B.

Arhitektura `ARH_KOMPARATOR` sadrži jedan proces u okviru kojeg se redom ispituju međusobni odnosi korespodentnih bita ulaznih brojeva počev od bita najveće važnosti. Ukoliko se utvrdi da su korespodentni biti različiti odmah se generišu izlazni signali. U slučaju da su svi bit ulaznih brojeva isti, na kraju lanca se ispituju vrednosti kaskadnih ulaza i u zavisnosti od njihove vrednosti se generišu izlazni signali.

Vremenski dijagram rada realizovanog komparatora prikazuje Slika 3.32.



Slika 3.32: Vremenski dijagram ponašanja realizovanog komparatora

Prethodna realizacija komparatora direktno odgovara realizaciji prema zadatoj tablici ulazno/izlaznih signala (Tabela 3.15), gde se porede korespodentni biti

ulaznih vektora redom od bita najveće važnosti do bita najmanje važnosti. Pomoću VHDL jezika za opis fizičke arhitekture, isti komparator se može realizovati na znatno jednostavniji način direktnim poređenjem kompletnih ulaznih vektora.

Takvu realizaciju prikazuje sledeći VHDL kod:

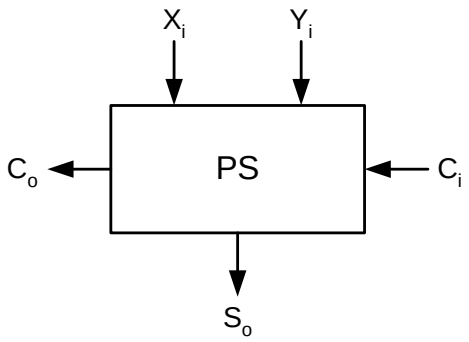
```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY KOMPARATOR IS PORT (
  -- ulazni brojevi
  iA, iB:        IN  std_logic_vector(3 DOWNTO 0);
  -- kaskadni ulazi
  iH3, iH2, iH1: IN  std_logic;
  -- izlazi komparatora
  oH3, oH2, oH1: OUT std_logic );
    -- H1 -> označava da je A=B
    -- H2 -> označava da je A<B
    -- H3 -> označava da je A>B
END KOMPARATOR;

ARCHITECTURE ARH_KOMPARATOR OF KOMPARATOR IS
BEGIN
  PROCESS (iA, iB, iH1, iH2, iH3) BEGIN
    IF    (iA > iB)   THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iA < iB)   THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSIF (iH3 = '1') THEN oH3 <= '1'; oH2 <= '0'; oH1 <= '0';
    ELSIF (iH2 = '1') THEN oH3 <= '0'; oH2 <= '1'; oH1 <= '0';
    ELSE                   oH3 <= '0'; oH2 <= '0'; oH1 <= '1';
    END IF;
  END PROCESS;
END ARH_KOMPARATOR;
```

Prethodni VHDL kod je funkcionalno potpuno ekvivalentan sa prvonavedenim kodom. Primećuje se da je prethodni kod mnogo čitljiviji i lakši za razumevanje i testiranje.

3.11 ZADATAK:

Izvršiti sintezu električne šeme punog sabirača.

Punim sabiračem naziva se kombinaciona mreža sa tri ulaza X_i, Y_i, C_i i dva izlaza S_o, C_o. koji predstavljaju vrednost zbira S_o i izlaznog prenosa C_o, ulaznih signala X_i i Y_i.



Slika 3.33: Ulazno/izlazni signali punog sabirača

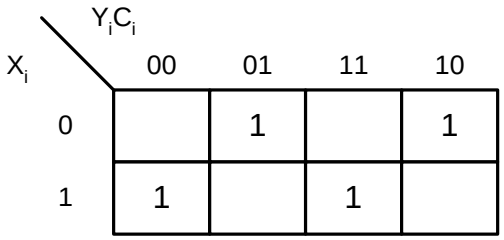
REŠENJE:

Na osnovu zahteva koji se postavljaju pred pun sabirač formira se prenosna tabela za ove funkcije (Tabela 3.16).

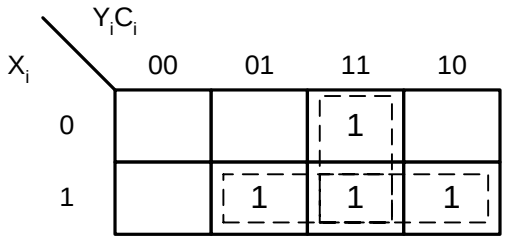
X_i	Y_i	C_i	S_o	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 3.16: Prenosna tabela punog sabirača

Minimizacijom pomoću Karnoovih karti se dobijaju prenosne funkcije punog sabirača, Slika 3.34 i Slika 3.35.



Slika 3.34: Karnoova karta za minimizaciju funkcije S_o



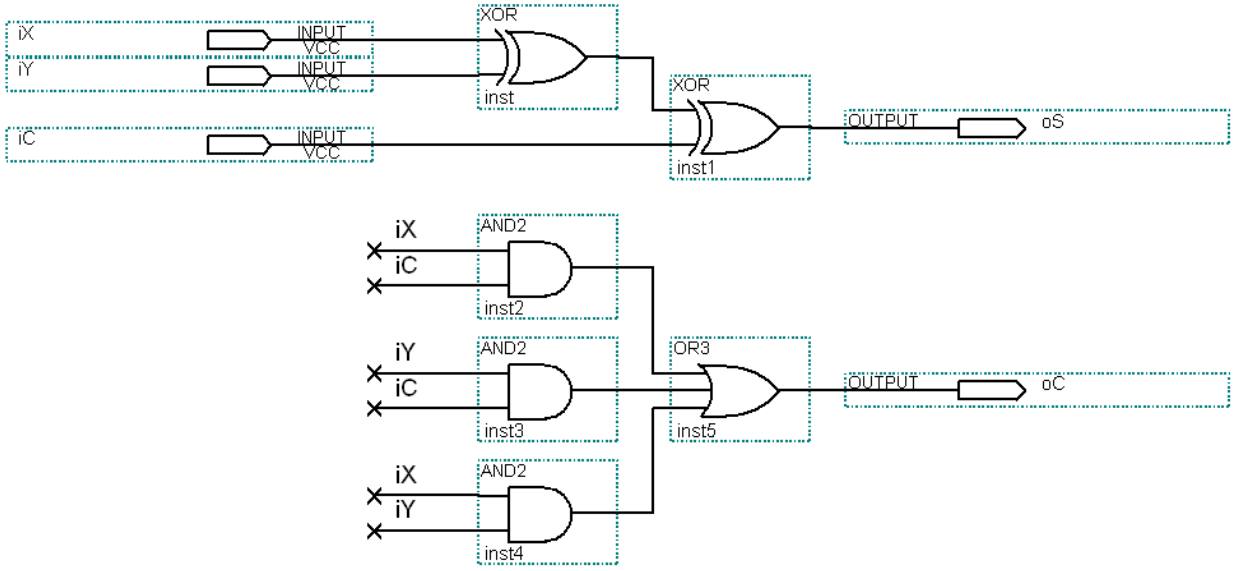
Slika 3.35: Karnoova karta za minimizaciju funkcije C_o

Prenosne funkcije punog sabirača (PS) su sledeće:

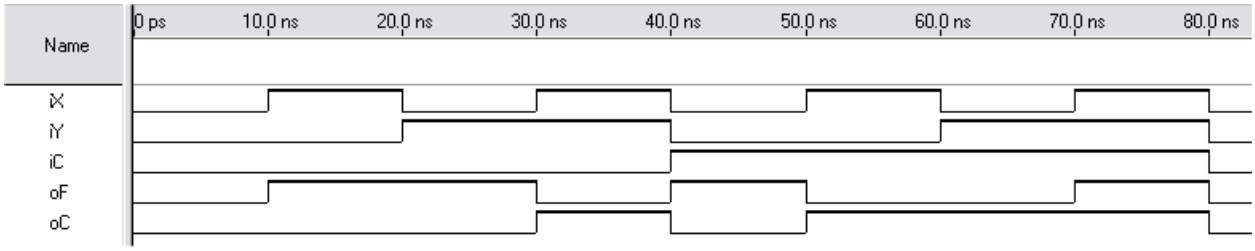
$$S_o = X_i \cdot \overline{Y_i} \cdot \overline{C_i} + \overline{X_i} \cdot \overline{Y_i} \cdot C_i + X_i \cdot Y_i \cdot C_i + \overline{X_i} \cdot Y_i \cdot \overline{C_i} = X_i \oplus Y_i \oplus C_i$$

$$C_o = X_i \cdot Y_i + X_i \cdot C_i + Y_i \cdot C_i$$

Slika 3.36 prikazuje realizaciju punog sabirača pomoću električne šeme. Simulaciju rada realizovane električne šeme prikazuje Slika 3.37.



Slika 3.36: Realizacija punog sabirača



Slika 3.37. Simulacija rada električne šeme punog sabirača

3.12 ZADATAK:

Realizovati kombinacionu mrežu koja će formirati bit parnosti P za četvorobitnu reč (D, C, B, A). Omogućiti odabir između parne i neparne parnosti. Realizovanu mrežu proširiti sa mogućnošću poređenja ulaznog bita parnosti P sa generisanim bitom parnosti za ulaznu reč. Rezultat poređenja prikazati izlaznim signalom greške G koji je na visokom nivou u slučaju da se pojavila greška.

REŠENJE:

Na osnovu ulaznih signala D, C, B i A treba formirati bit parnosti P. U slučaju generisanja bita parne parnosti potrebno je da u petobitnoj reči koja se sastoji od ulaznih signala D, C, B, A i generisanog bita parne parnosti P_p bude sadržan paran broj binarnih jedinica. Kod neparne parnosti P_N ukupan broj binarnih jedinica je neparan. U skladu sa time formira se kombinaciona tabela ulaza/izlaza prilikom formiranja bita parnosti, Tabela 3.17.

Iz tabele se vidi da su dobijeni biti parne i neparne parnosti komplementarni. Stoga je dovoljno realizovati kombinacionu mrežu za dobijanje bita parne parnosti P_P i sa jednim invertorom formirati bit neparne parnosti P_N. Na izlaz se prosleđuje

jedna od ove dve vrednosti u zavisnosti od stanja ulaznog signala za odabir parne i neparne parnosti $E/\overline{O}$.

D	C	B	A	P_P	P_N
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Tabela 3.17: Tabela formiranja bita parnosti

Tabela 3.18 prikazuje zavisnost izlaznog signala parnosti u zavisnosti od generisanog signala parne parnosti i ulaznog signala za odabir parne i neparne parnosti.

$E/\overline{O} = 1$ parna parnost

$E/\overline{O} = 0$ neparna parnost

P_P	$E/\overline{O}$	P
0	0	1
0	1	0
1	0	0
1	1	1

Tabela 3.18: Formiranje izlaznog signala parnosti

Na osnovu tabele formiranja bita parnosti (Tabela 3.17) dobija se prenosna funkcija za dobijanje bita parne parnosti: $P_P = A \oplus B \oplus C \oplus D$

Tabela 3.18 prikazuje zavisnost generisanog signala parne parnosti i ulaznog signala za odabir parne i neparne parnosti. Na osnovu tabele dobija se prenosna

funkcija $P = \overline{P_P \oplus E/\overline{O}}$. Formiranje logičke šeme za generisanje izlaznog bita parnosti se može realizovati preko dobijene prenosne funkcije ili jednostavno sa multiplekserom 2×1 koji u zavisnosti od stanja ulaznog signala $E/\overline{O}$ na izlaz propušta bit parne parnosti ili bit neparne parnosti.

Za potrebe poređenja generisanog bita parnosti i ulaznog bita parnosti formira se Tabela 3.19.

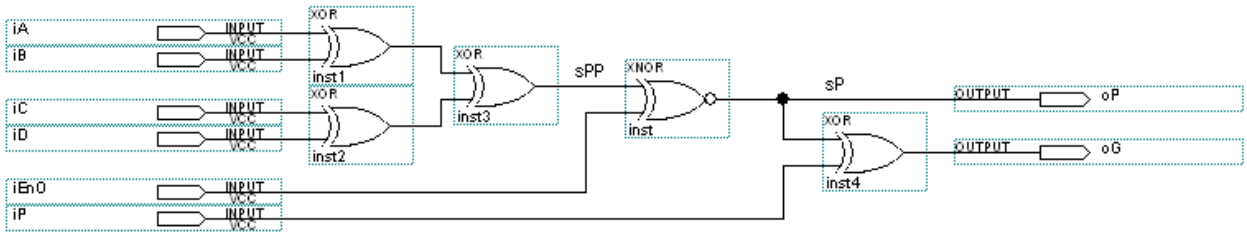
G = 0 greška ne postoji
G = 1 greška postoji

P	P_{UL}	G
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 3.19: Tabela formiranja signala greške

Iz tabele se dobija prenosna funkcija $G = P \oplus P_{UL}$ koja opisuje izlazni signal indikacije greške parnosti u primljenoj reči.

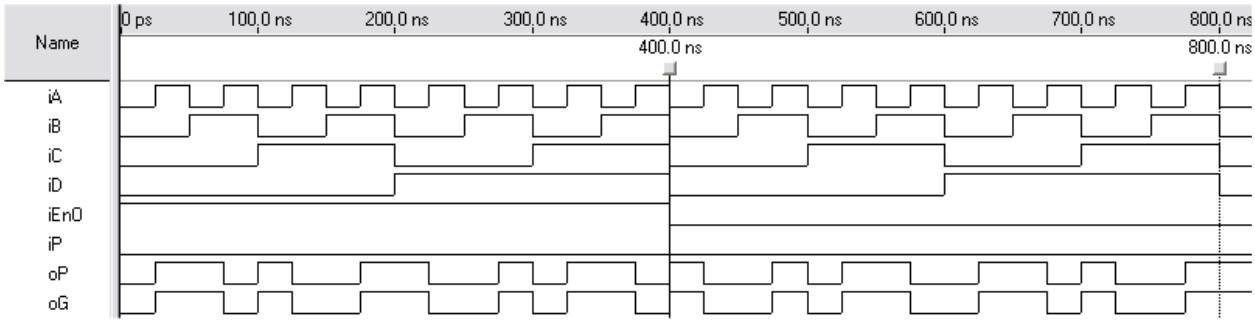
Na osnovu prethodne analize i dobijenih prenosnih funkcija pristupa se formiranju logičke šeme elemenata tražene kombinacione mreže, Slika 3.38.



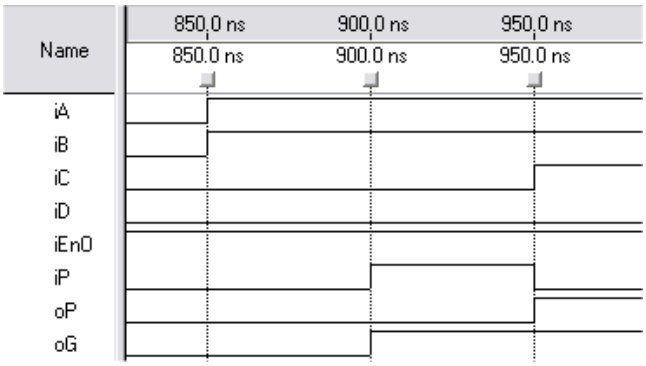
Slika 3.38: Logička šema realizovane kombinacione mreže

Slika 3.39 prikazuje vremenski dijagram generisanja bita parnosti. U vremenskom intervalu od 0 do 400ns prikazano je generisanje bita parne parnosti (`iEnO=1`) za sve vrednosti ulaznog vektora `iA,iB,iC,iD`. U nastavku vremenskog dijagrama, od 400ns do 800ns, je prikazano generisanje bita neparne parnosti (`iEnO=0`). Vidi se da su generisane vrednosti na izlaznom signalu oP u skladu sa vrednostima koje sadrži Tabela 3.17.

Slika 3.40 prikazuje generisanje signala greške parnosti za tri kombinacije ulaznih vrednosti.



Slika 3.39: Generisanje bita parne i neparne parnosti

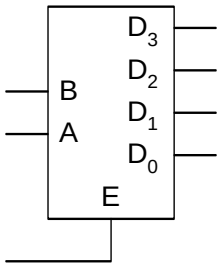


Slika 3.40: Generisanje signala greške parnosti

Prva situacija, t=(850ns÷900ns), je u slučaju kada se generisani bit parnosti `oP` za ulazni vektor (`iA,iB,iC,iD=1100`) podudara sa primljenim bitom parnosti `iP`. U tom slučaju je signal greške parnosti `oG` na nuli. Drugi slučaj, t=(900ns÷950ns), prikazuje situaciju kada je za isti ulazni vektor primljen drugi bit parnosti. Pošto se razlikuju primljeni i generisani bit parnosti, generiše se signal greške parnosti `oG=1`. Poslednja prikazana kombinacija ulaznih signala, t=(950ns÷1000ns), prikazuje situaciju kada se javila greška u prijemu ulazne reči. Naime, primljena je pogrešna vrednost ulaznog signala `iC` pa se zbog toga javlja razlika u generisanom i primljenom bitu parnosti, zbog čega se generiše greška parnosti

3.13 ZADATAK:

Slika 3.41 prikazuje blok šemu dekodera 2×4. Potrebno je formirati dekoder 3×8 pomoću dva dekodera 2×4. Prikazati blok šemu sa odgovarajućim oznakama ulaznih i izlaznih signala i navesti tabelu ulaza/izlaza formiranog dekodera.



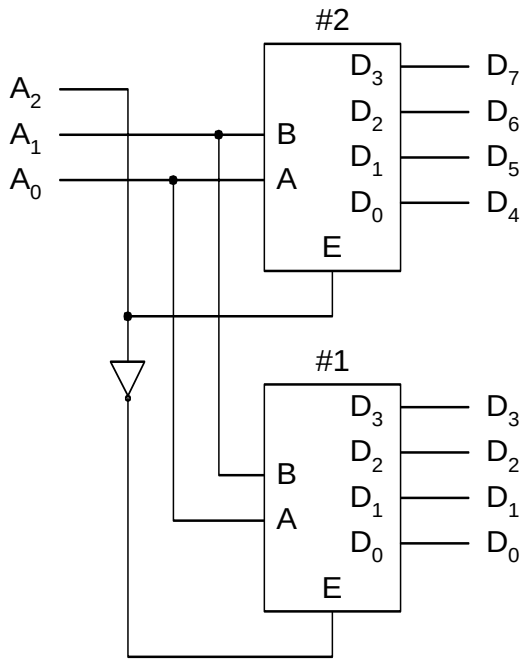
Slika 3.41: Dekoder 2×4

REŠENJE:

Dat je dekoder 2×4 sa ulazom dozvole dekodovanja E koji je aktivan na visokom nivou. Ulazni signali su A (bit manje važnosti) i B (bit veće važnosti). Ulazni signali dekodera 3×8 će se označiti sa A_0, A_1 i A_2, gde je A_0 bit najmanje važnosti, a A_2 bit najveće važnosti.

Traženi dekoder 3×8 će se formirati tako što će se ulazni signal A_0 dovesti na ulaz A oba dekodera, a ulazni signal A_1 dovesti na ulaz B oba dekodera. Na ovaj način je obezbeđeno da oba dekodera 2×4 dekodiraju prve četiri vrednosti ulaznog vektora A=(A_0,A_1,A_2). Cilj je da se sa ulaznim signalom A_2 razdvoji dekodovanje prve četiri vrednosti ulaznog vektora A (gde je A_2=0) i druge četiri vrednosti ulaznog vektora A (gde je A_2=1) na dva dekodera. To se realizuje tako što se invertovana vrednost ulaznog signala A_2 dovede na ulaz dozvole dekodovanja prvog dekodera, a njegova neinvertovana vrednost na ulaz dozvole dekodovanja drugog dekodera. Time će prvi dekoder biti aktivan kada je A_2=0 i na njegovim izlazima će se generisati signali koji odgovaraju vrednostima 0, 1, 2 i 3 ulaznog vektora A. Slično, drugi dekoder će biti aktivan kada je A_2=1 i na njegovim izlazima će se generisati signali koji odgovaraju vrednostima 4, 5, 6 i 7 ulaznog vektora A.

Slika 3.42 prikazuje blok šemu dekodera 3×8 formiranog pomoću dva dekodera 2×4. Na slici su prvi i drugi dekoder označeni redom sa #1 i #2.



Slika 3.42: Dekoder 3×8 formiran pomoću dekodera 2×4

Tabelu ulazno/izlaznih vrednosti ovako formiranog dekodera 3×8 prikazuje Tabela 3.20.

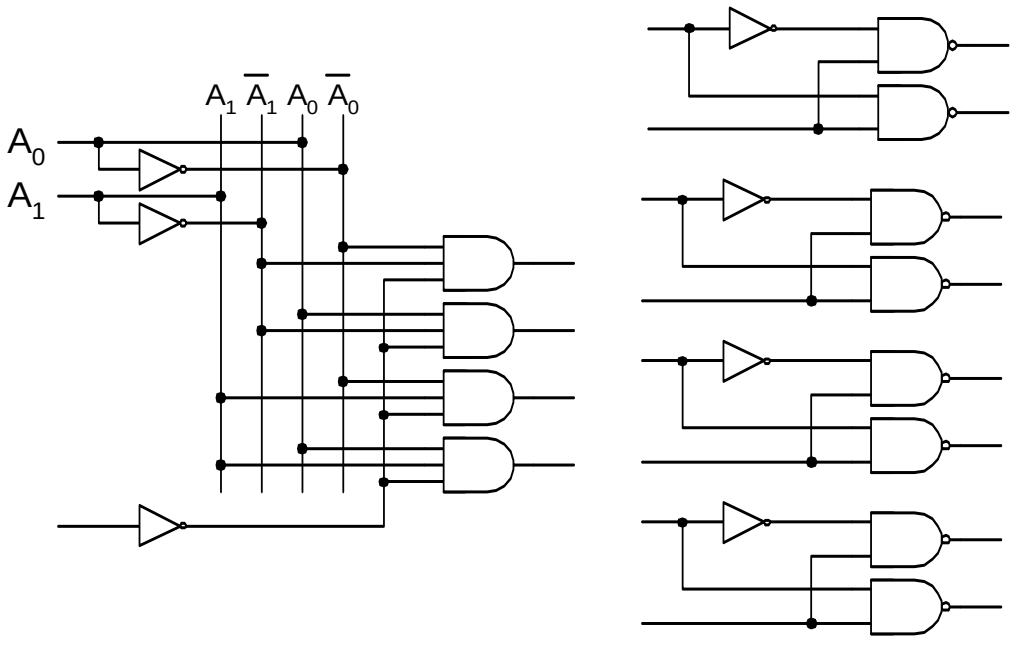
Ulazi			Izlazi								Aktivan dekoder
A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	0	0	1	#1
0	0	1	0	0	0	0	0	0	1	0	#1
0	1	0	0	0	0	0	0	1	0	0	#1
0	1	1	0	0	0	0	1	0	0	0	#1
1	0	0	0	0	0	1	0	0	0	0	#2
1	0	1	0	0	1	0	0	0	0	0	#2
1	1	0	0	1	0	0	0	0	0	0	#2
1	1	1	1	0	0	0	0	0	0	0	#2

Tabela 3.20: Tabela ulaza/izlaza formiranog dekodera 3×8

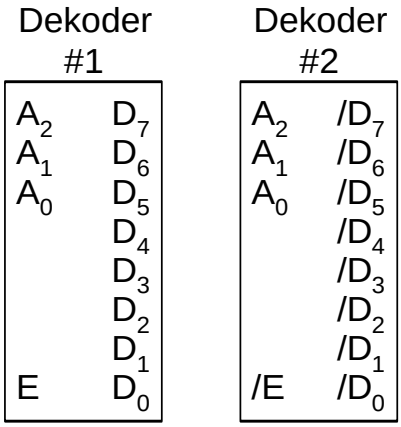
3.14 ZADATAK:

Slika 3.43 prikazuje logičku šemu dekodera 2×4 i logičke šeme četiri identična dekodera 1×2. Potrebno ih je povezati tako da čine dekoder 3×8 koji je funkcionalno ekvivalentan dekoderu #1 ili dekoderu #2 koje prikazuje Slika 3.44. Odabrati jednostavnije rešenje i navesti oznake ulaznih i izlaznih signala.

Signali koji su aktivni na niskom logičkom nivou su označeni znakom / ispred imena signala.



Slika 3.43: Logičke šeme dekodera 2×4 i 1×2



Slika 3.44: Sprežni signali dekodera koji treba formirati

REŠENJE:

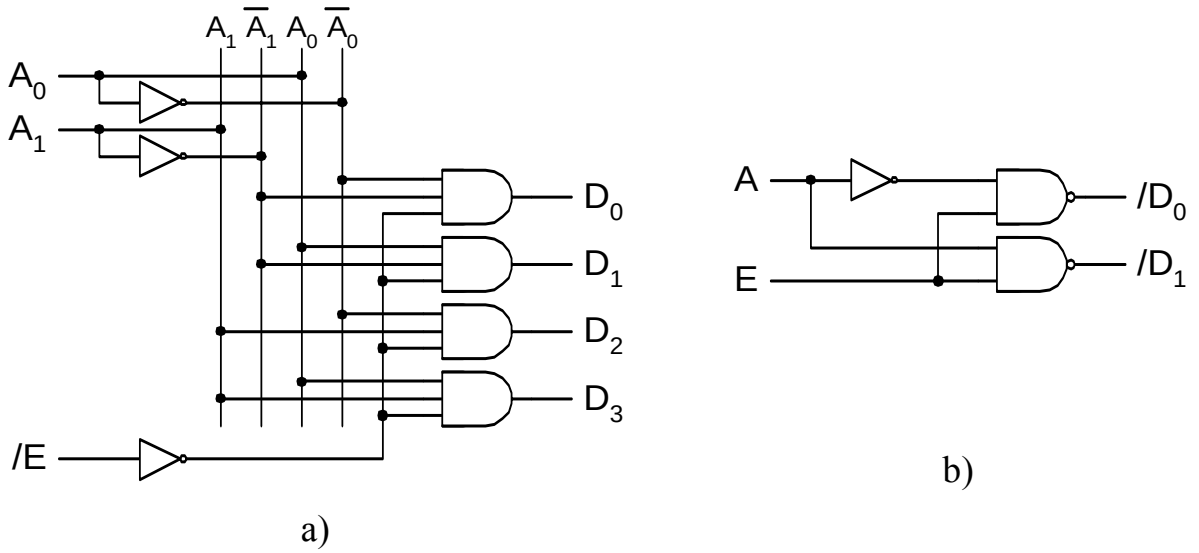
Prvo je potrebno izvršiti analizu rada datih logičkih šema.

Na logičkoj šemi dekodera 2×4 su navedene oznake ulaznog vektora $A=(A_0,A_1)$, gde je A_0 bit manje važnosti, a A_1 bit veće važnosti. Vidi se da su izlazna troulazna I kola aktivna samo ako treći, neoznačeni, ulazni signal ima vrednost nula. Na osnovu toga se zaključuje da taj ulazni signal predstavlja signal dozvole dekodovanja koji je aktivan na niskom logičkom nivou. Sledi određivanje pozicija izlaznih signala D_0, D_1, D_2 i D_3. Pošto su na prvo I kolo dovedene invertovane vrednosti ulaznih signala A_0 i A_1, sledi da izlaz ovog logičkog kola generiše izlazni signal D_0. Drugo I kolo generiše izlazni signal D_1 pošto su na njegove ulaze dovedeni ulazni signal A_0 i invertovana vrednost ulaznog signala A_1. Time izlaz ovog logičkog kola ima vrednost jedan samo ako su $A_1=0$, $A_0=1$ i aktivan je signal dozvole dekodovanja (/E=0). Slično, na izlaz trećeg logičkog kola ima vrednost jedan samo ako su $A_1=1$, $A_0=0$ i aktivan je signal dozvole dekodovanja (/E=0), čime se generiše izlazni signal D_2. Treće logičko kolo generiše izlazni signal D_3 (izlaz je aktivan samo ako su $A_1=1$, $A_0=1$ i /E=0).

Slika 3.45 a) prikazuje logičku šemu dekodera 2×4 sa označenim svim ulaznim i izlaznim signalima.

Dati dekoder 1×2 je realizovan pomoću NI logičkih kola. Na taj način je vrednost izlaznih signala u invertovanoj logici, tj. izlazni signal je aktivan ako je njegova vrednost jednaka 0. Od dva ulazna signala, donji je direkto doveden na ulaze oba logička kola na osnovu čega se zaključuje da on predstavlja signal dozvole dekodovanja, označen sa E, aktivan na visokom logičkom nivou. Drugi, gornji, ulazni signal (označen sa A) se direktno dovodi na drugo NI kolo, a njegova invertovana vrednost na prvo NI kolo. Izlaz prvog NI kola je aktivan, ima vrednost nula, samo ako je A=0 i E=1, dok je izlaz drugog NI kola aktivan ako je A=1 i E=1. Na osnovu toga se zaključuje da prvo logičko kolo generiše izlazni signal $/D_0$, a drugo logičko kolo izlaz $/D_1$.

Slika 3.45 b) prikazuje logičku šemu dekodera 1×2 sa označenim svim ulaznim i izlaznim signalima.



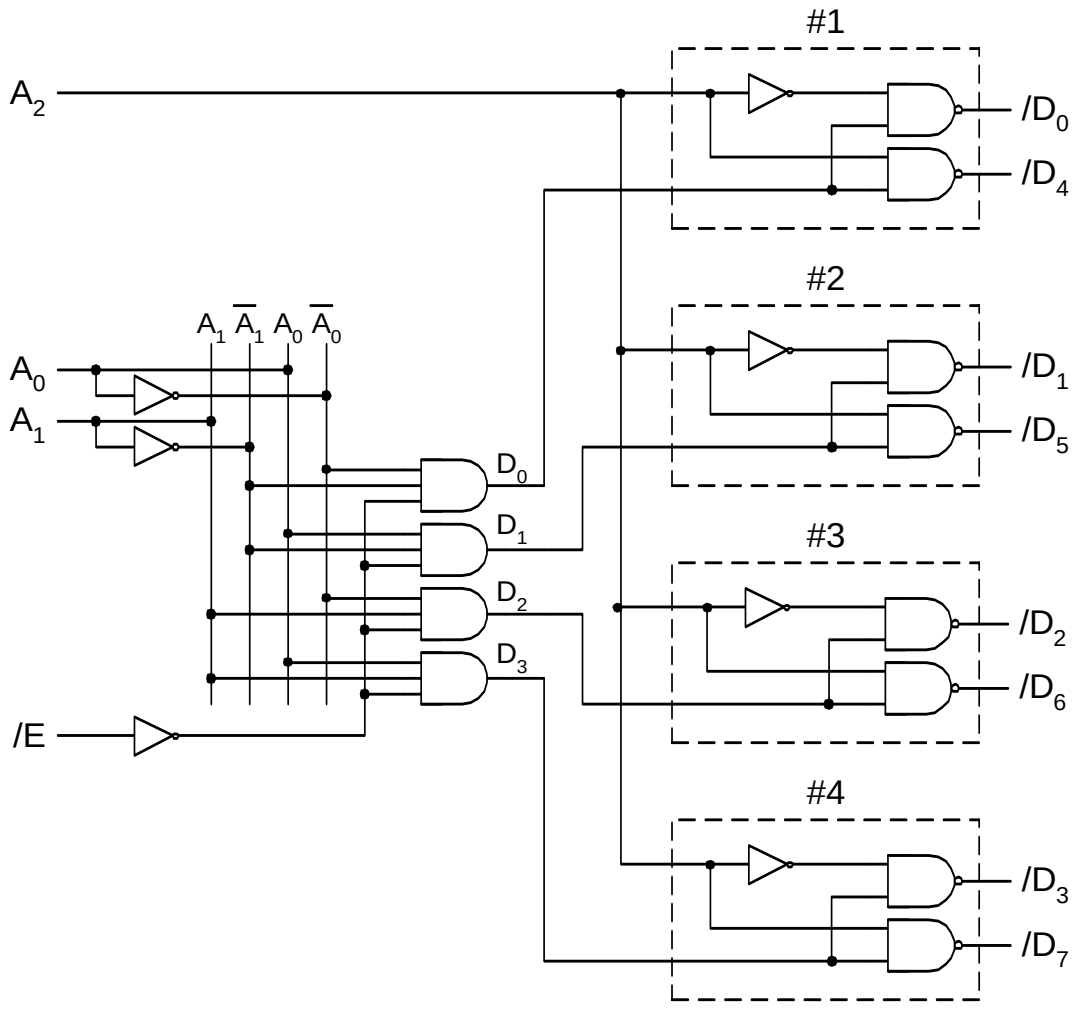
Slika 3.45: a) Sprežni signali dekodera 2×4
b)Sprežni signali dekodera 1×2

Nakon analize rada datih dekodera sledi njihovo povezivanje sa ciljem dobijanja dekodera 3×8. Pošto je ulazni signal dozvole dekodovanja dekodera 2×4 aktivan na niskom nivou, a dati dekoder 1×2 generiše izlazne signale aktivne na niskom logičkom nivou, sledi da je lakše formirati dekoder 3×8 koji je funkcionalno ekvivalentan dekoderu #2 (Slika 3.44).

Signali A_0 i A_1 ulaznog vektora $A=(A_0,A_1,A_2)$ se dovode na odgovarajuće ulaze dekodera 2×4, kao i signal dozvole dekodovanja /E. Na ovaj način je obezbeđeno da dekoder 2×4 dekodira prve četiri vrednosti ulaznog vektora $A=(A_0,A_1,A_2)$. Cilj je da se pomoću četiri dekodera 1×2 i ulaznog signala A_2 razdvoji dekodovanje prve četiri vrednosti ulaznog vektora A (gde je $A_2=0$) i druge četiri vrednosti ulaznog vektora A (gde je $A_2=1$). To se postiže tako što se ulazni signal A_2 dovodi na ulaz A sva četiri dekodera 1×2. Na taj način će izlaz $/D_0$ sva četiri dekodera 1×2 biti aktivan kada je $A_2=0$, dok će izlaz $/D_1$ biti aktivan kada je $A_2=1$. Povezivanjem izlaza D_0 dekodera 2×4 na ulazni signal dozvole dekodovanja E prvog dekodera 1×2 dobija se da će njegov izlaz $/D_0$ biti aktivan kada je $A_2=0$, $A_1=0$ i $A_0=0$, dok će izlaz $/D_1$ biti aktivan kada je $A_2=1$, $A_1=0$ i $A_0=0$. Na taj način su formirani izlazi $/D_0$ i $/D_4$ traženog dekodera 3×8.

Slično, povezivanjem izlaza D_1 dekodera 2×4 na ulazni signal dozvole dekodovanja drugog dekodera 1×2 formiraju se izlazi $/D_1$ i $/D_5$ dekodera 3×8. Izlazi $/D_2$ i $/D_6$ dekodera 3×8 se formiraju povezivanjem izlaza D_2 dekodera 2×4 na ulazni signal dozvole dekodovanja trećeg dekodera 1×2, dok se izlazi $/D_3$ i $/D_7$ dekodera 3×8 formiraju pomoću četvrtog dekodera 1×2 tako što se na ulazni signal dozvole dekodovanja povezuje izlaz D_3 dekodera 2×4.

Slika 3.46 prikazuje blok šemu dekodera 3×8 formiranog pomoću jednog dekodera 2×4 i četiri dekodera 1×2. Na slici su prvi, drugi, treći i četvrti dekoder 1×2 označeni redom sa #1, #2, #3 i #4.



Slika 3.46: Blok šema formiranog dekodera 3×8

3.15 ZADATAK:

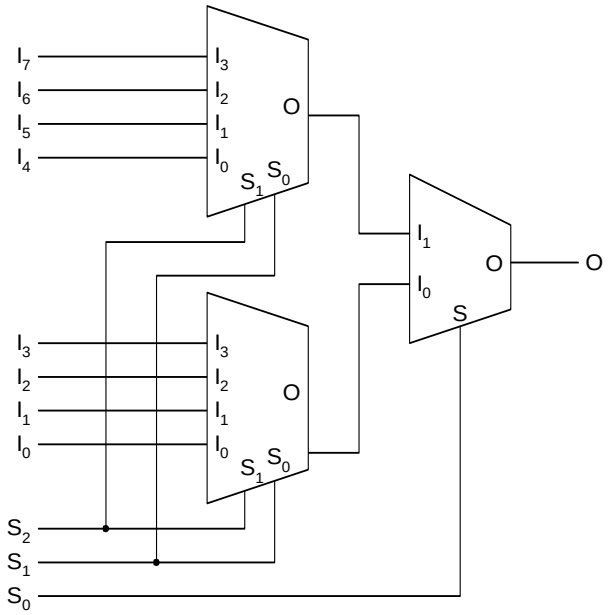
Potrebno je formirati multiplekser 8×1 pomoću dva multipleksera 4×1 i jednog multipleksera 2×1. Prikazati blok šemu sa odgovarajućim oznakama ulaznih i izlaznih signala i navesti tabelu ulaza/izlaza formiranog multipleksera.

REŠENJE:

Multiplekser 8×1 se formira kaskadnom vezom gde su u prvom stepenu multiplekseri 4×1, a njihovi izlazi su spojeni na ulaze multipleksera 2×1 koji se nalazi u drugom stepenu. Adresni ulaz najveće važnosti, označen sa S_2, multipleksera 8×1 se dovodi na adresni ulaz multipleksera u drugom stepenu, dok se preostala dva bita adresnog ulaza multipleksera 8×1, tj. biti S_1 i S_0, dovode na adresne ulaze oba multipleksera u prvom stepenu. Na taj način je obezbeđeno da

adresni ulaz S_2, multiplesera 8×1 razdvaja ulazne signale na dve grupe po četiri signala. Ulazi $I_0 \div I_7$ se dovode na ulaze multipleksera prvog stepena tako što se ulazi $I_0 \div I_3$ dovode na ulaze multipleksera čiji je izlaz spojen na ulaz multipleksera 2×1 sa adresom nula, dok se ulazi $I_4 \div I_7$ dovode na ulaze drugog multipleksera čiji je izlaz spojen na ulaz multipleksera 2×1 sa adresom jedan.

Slika 3.47 prikazuje kaskadnu vezu multipleksera koja čini multiplekser 8×1, dok tabelu ulazno/izlaznih vrednosti ovako formiranog multipleksera 8×1 prikazuje Tabela 3.21.



Slika 3.47: Multiplekser 8×1

Adresni ulazi			Izlaz
S_2	S_1	S_0	O
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Tabela 3.21: Tabela ulaza/izlaza formiranog multipleksera 8×1

3.16 ZADATAK:

Data je funkcija F (Tabela 3.22).

a) Implementirati funkciju F pomoću multipleksera 8×1
b) Implementirati funkciju F pomoću multipleksera 4×1

A	B	C	F
0	0	0	1
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

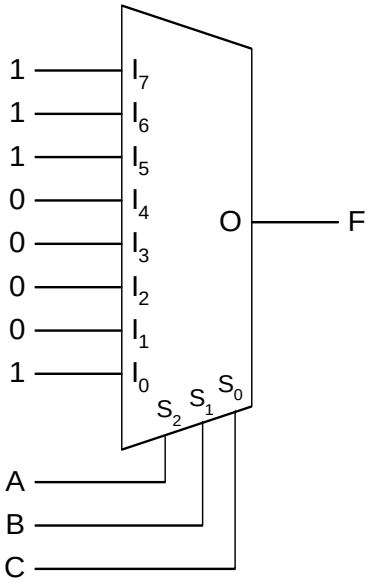
Tabela 3.22: Tabela istinitosti funkcije F

REŠENJE:

a)

Implementacija zadate funkcije pomoću multipleksera 8×1 je trivijalna pošto je funkcija F funkcija od tri promenljive (A, B i C), a multiplekser 8×1 poseduje upravo tri adresna ulaza. Stoga je dovoljno ulazne promenljive dovesti na adresne ulaze multipleksera, vodeći računa o poziciji bita u adresnoj reči (A je bit najveće važnosti, a C bit najmanje važnosti), i na ulaz sa odgovarajućom adresom dovesti vrednost funkcije, tj. 1 ili 0.

Slika 3.48 prikazuje implementaciju funkcije F pomoću multipleksera 8×1



Slika 3.48: Funkcija F formirana pomoću multipleksera 8×1

U fizičkoj realizaciji tamo gde funkcija F ima vrednost 1 dovodi se napajanje VCC, dok se ulazi označeni sa 0 spajaju na uzemljenje GND.

b)

Pošto multiplekser 4×1 ima samo dva adresna ulaza potrebno je formirati novu tabelu koja funkciju F opisuje u zavisnosti od npr. promenljivih A i B. To se postiže grupisanjem vrednosti funkcije F za jednake vrednosti ulaznih promenljivih A i B. Na taj način funkcija F umesto dve vrednosti, 0 ili 1, može imati sledeće četiri vrednosti 0, 1, C ili $\overline{C}$.

Formiranje nove tabele istinitosti funkcije F prikazuje Tabela 3.23.

A	B	C	F	
0	0	0	1	$F = \overline{C}$
0	0	1	0	
0	1	0	0	$F = 0$
0	1	1	0	
1	0	0	0	$F = C$
1	0	1	1	
1	1	0	1	$F = 1$
1	1	1	1	

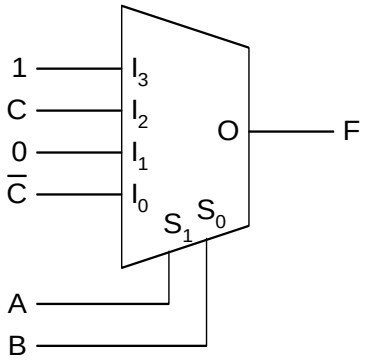
$\Leftrightarrow$

A	B	F
0	0	$\overline{C}$
0	1	0
1	0	C
1	1	1

Tabela 3.23: Skraćena tabela istinitosti funkcije F

Sada se funkcija F formira na isti način kao i u zadatku pod a). Na adresne ulaze multipleksera 4×1 se dovode ulazne promenljive A i B i na ulaz sa odgovarajućom adresom se dovodi vrednost funkcije koja u ovom slučaju može biti 0, 1, C ili $\overline{C}$.

Slika 3.49 prikazuje implementaciju funkcije F pomoću multipleksera 4×1



Slika 3.49: Funkcija F formirana pomoću multipleksera 4×1

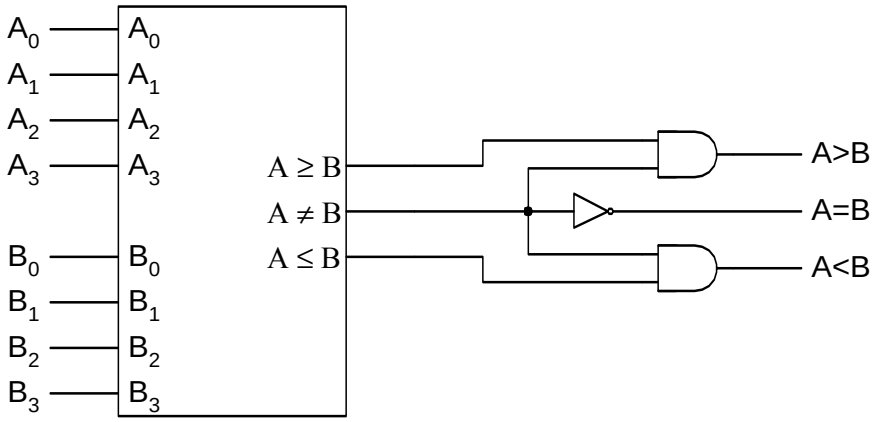
3.17 ZADATAK:

Dat je komparator četvorobitnih brojeva A i B koji generiše tri izlaza: $A \geq B$, $A \leq B$ i $A \neq B$. Nacrtati logičku šemu sa takvim komparatorom i svom potrebnom logikom radi formiranja komparatora sa standardnim izlazima $A > B$, $A < B$ i $A = B$.

REŠENJE:

Izlazni signal $A = B$ se jednostavno formira invertovanjem izlaznog signala $A \neq B$ datog komparatora. Ulazni broj A je veći od ulaznog broja B ako su aktivni signali $A \geq B$ i $A \neq B$. Iz ovog sledi da se izlazni signal $A > B$ formira pomoću dvoulaznog I kola na čije ulaze se dovode izlazi $A \geq B$ i $A \neq B$ datog komparatora. Slično, izlazni signal $A < B$ se formira pomoću dvoulaznog I kola na čije ulaze se dovode izlazi $A \leq B$ i $A \neq B$ datog komparatora.

Slika 3.50 prikazuje odgovarajuću logičku šemu.



Slika 3.50: Komparator sa standardnim izlazima $A > B$, $A < B$ i $A = B$

3.18 ZADATAK:

a) Pomoću dva četvorobitna komparatora (74LS85) i potrebnih logičkih kola odrediti da li četvorobitni broj P=(P_3, P_2, P_1, P_0) zadovoljava uslov $4 \le P < 8$.

b) Isprojektovati minimalnu kombinacionu mrežu koja proverava da li četvorobitni broj P=(P_3, P_2, P_1, P_0) zadovoljava uslov $4 \le P < 8$. Minimizaciju realizovati pomoću Karnoovih karti.

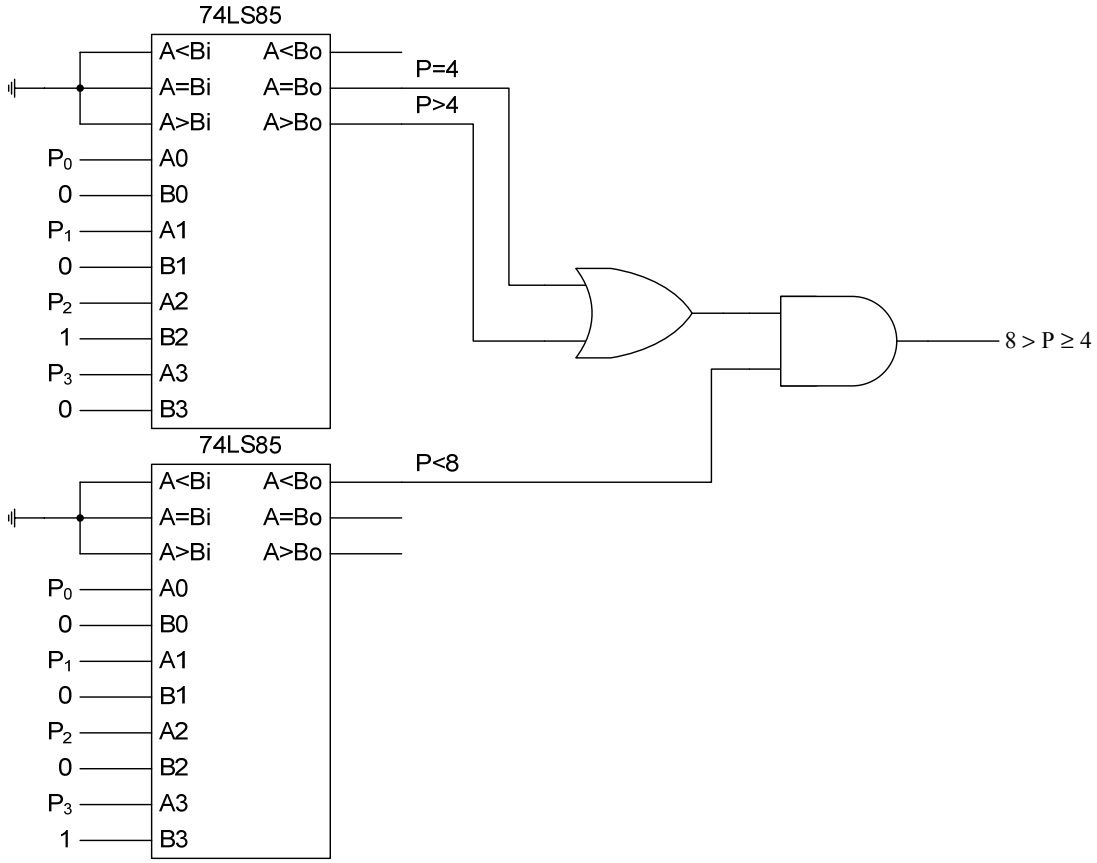
REŠENJE:

a)

Provera zadatog uslova se može realizovati na sledeći način. Pomoću prvog komparatora poredi se broj P sa brojem 4, dok se sa drugim komparatorom broj P poredi sa brojem 8. Neka se broj P dovede na ulaz A oba komparatora, a zadati brojevi na ulaz B. Kaskadni ulazi oba komparatora $A > B$, $A < B$ i $A = B$ se spoje na masu, tj. logičku nulu, pošto se ne koriste prilikom poređenja.

Ako je zadati uslov zadovoljen moguće su tri situacije:

1. Aktivan je izlaz $A = B$ prvog komparatora koji odgovara slučaju P=4, ili
2. Aktivan je izlaz $A > B$ prvog komparatora koji odgovara slučaju P>4, i
3. Aktivan je izlaz $A < B$ drugog komparatora koji odgovara slučaju P<8



Slika 3.51: Provera uslova $4 \le P < 8$ pomoću komparatora

Iz ovog sledi da je za generisanje izlaznog signala koji ukazuje ispunjenje uslova $4 \le P < 8$ potrebno iskoristi jedno troulazno ILI kolo na čije ulaze se dovedu prethodna tri izlaza oba komparatora.

Slika 3.51 prikazuje odgovarajuću logičku šemu za realaciju zadate kombinacione mreže.

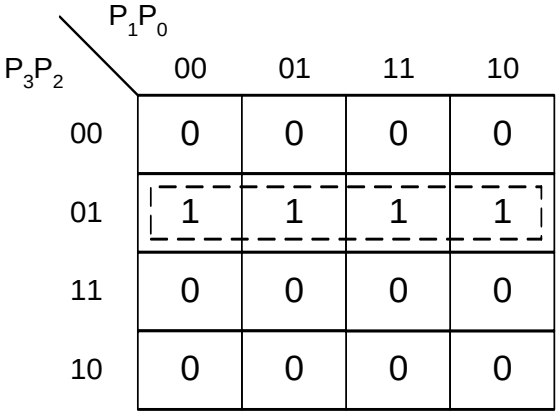
b)

Za dobijanje potrebne minimalne kombinacione mreže koja ukazuje da li je zadoveljen uslov $4 \le P < 8$ prvo je potrebno formirati kombinacionu tabelu tražene funkcije, Tabela 3.24.

Na osnovu kombinacione tabele se popunjava Karnoova karta radi određivanja minimalne forme funkcije, Slika 3.52.

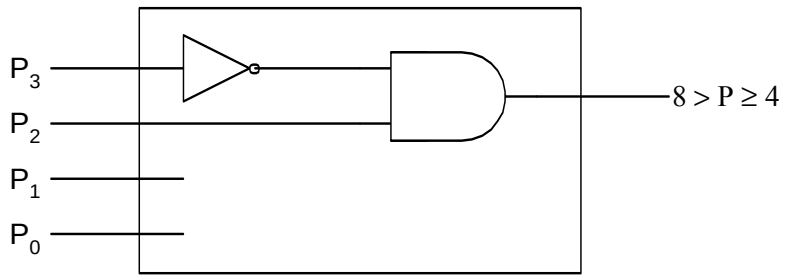
P_3	P_2	P_1	P_2	$4 \le P < 8$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Tabela 3.24: Kombinaciona tabela



Slika 3.52: Karnoova karta

Na osnovu Karnoove se dobija funkcija $\overline{P_3} \cdot P_2$, na osnovu koje se formira minimalna kombinaciona mreža, Slika 3.53.



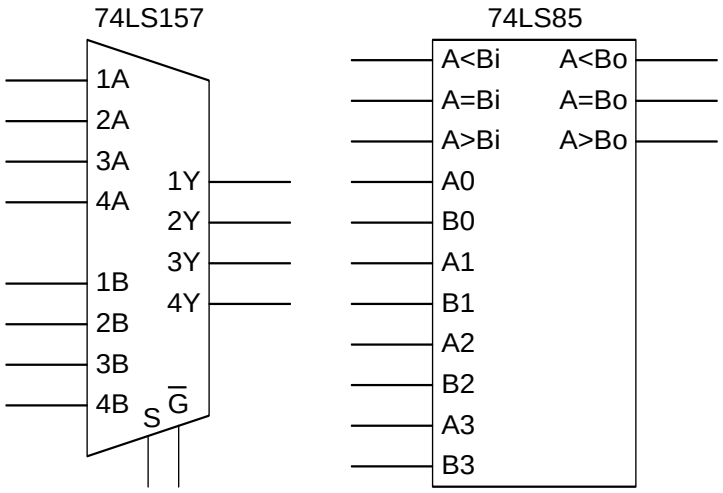
Slika 3.53: Minimalna kombinaciona mreža za proveru uslova $4 \le P < 8$

3.19 ZADATAK:

a) Projektovati kombinacionu mrežu za određivanje najvećeg broja od tri četvorobitna broja P=(P_3, P_2, P_1, P_0), Q=(Q_3, Q_2, Q_1, Q_0) i R=(R_3, R_2, R_1, R_0) koristeći dva multipleksera 74LS157 i dva komparatora 74LS85. Izlaz označiti sa L=(L_3, L_2, L_1, L_0).

b) Ako su P=5=(0101), Q=12=(1100) i R=9=(1001) tada je najveći broj Q, pa sledi da L treba da postane 12=(1100). Označiti vrednosti signala na formiranoj logičkoj šemi.

Slika 3.54 prikazuje blok šeme multipleksera 74LS157 i komparatora 74LS85.



Slika 3.54: Blok šeme multiplesera 74LS157 i komparatora 74LS85

REŠENJE:

a)

Tražena kombinaciona mreža se formira tako što se prvo porede proizvoljna dva od tri zadata broja i odabira se veći broj. Zatim se odabrani broj poredi sa trećim zadatim brojem i odabirom većeg se generiše vrednost izlaznog, najvećeg, broja.

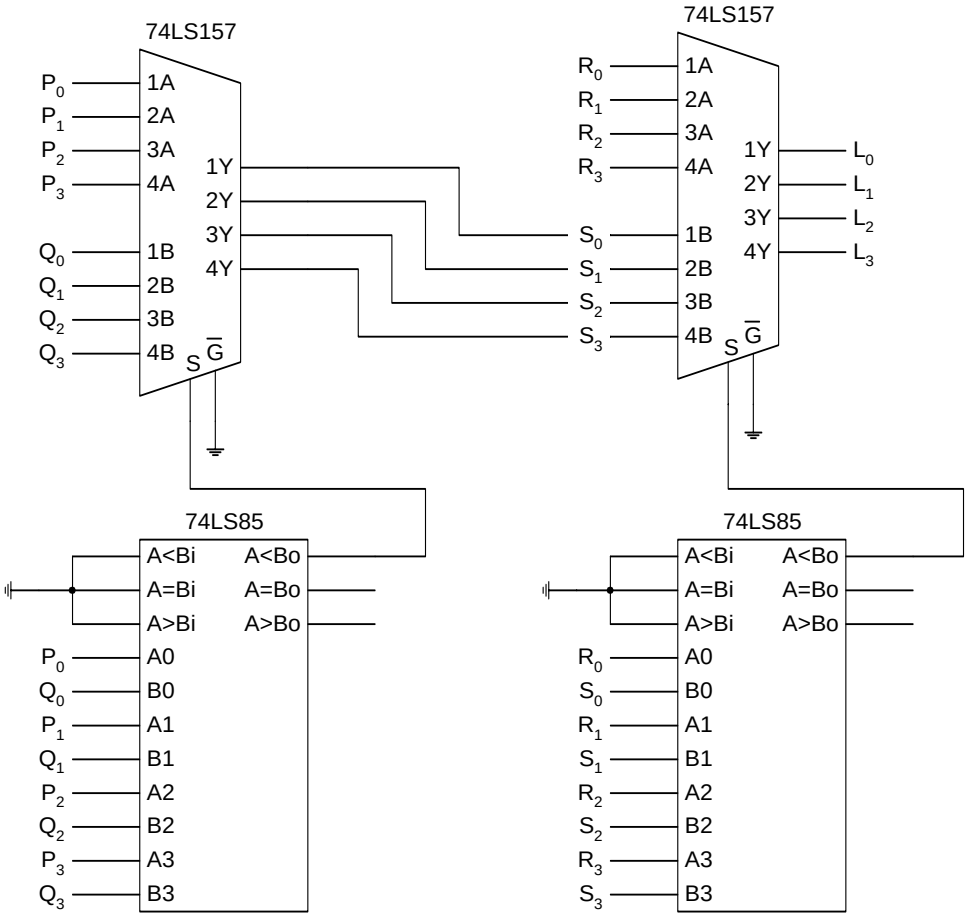
Neka se prvo porede brojevi P i Q tako što se broj P dovede na ulaz A prvog komparatora a broj Q se dovede na ulaz B. Kaskadni ulazi $A > B$, $A < B$ i $A = B$ se spoje na masu, tj. logičku nulu, pošto se ne koriste prilikom ovog poređenja. Izlazni signali $A > B$, $A < B$ i $A = B$ komparatora ukazuju na međusobni odnos brojeva P i Q. Treba naglasiti da su izlazni signali komparatora 74LS85 aktivni na visokom logičkom nivou.

Odabir većeg broja se vrši pomoću multipleksera 74LS157. Neka se i kod multipleksera na ulaz A dovede broj P a na ulaz B broj Q. Ulaz dozvole multipleksiranja $\overline{G}$ se spaja na masu čime se dozvoljava multipleksiranje signala. Potrebno je pomoću adresnog ulaza S multipleksera odabrati veći od brojeva P i Q. U slučaju da je S=1 odabira se broj Q pošto je on doveden na ulaz B. Ovo je bitno iz razloga što su izlazi komparatora aktivni na visokom nivou.

Stoga, za slučaj $Q > P$ treba obezbediti da adresni ulaz multipleksera bude na visokom logičkom nivou. Pošto je broj Q doveden na ulaz B komparatora, tada je za $Q > P$ aktivan izlazni signal $A < B$. Odavde sledi da na adresni ulaz S multipleksera treba dovesti izlaz komparatora $A < B$.

U slučaju da je $Q < P$ vrednost odabranog izlaznog signala $A < B$ komparatora će biti 0, čime će se odabrati broj P pošto je on doveden na ulaz A multipleksera čija je adresa upravo 0.

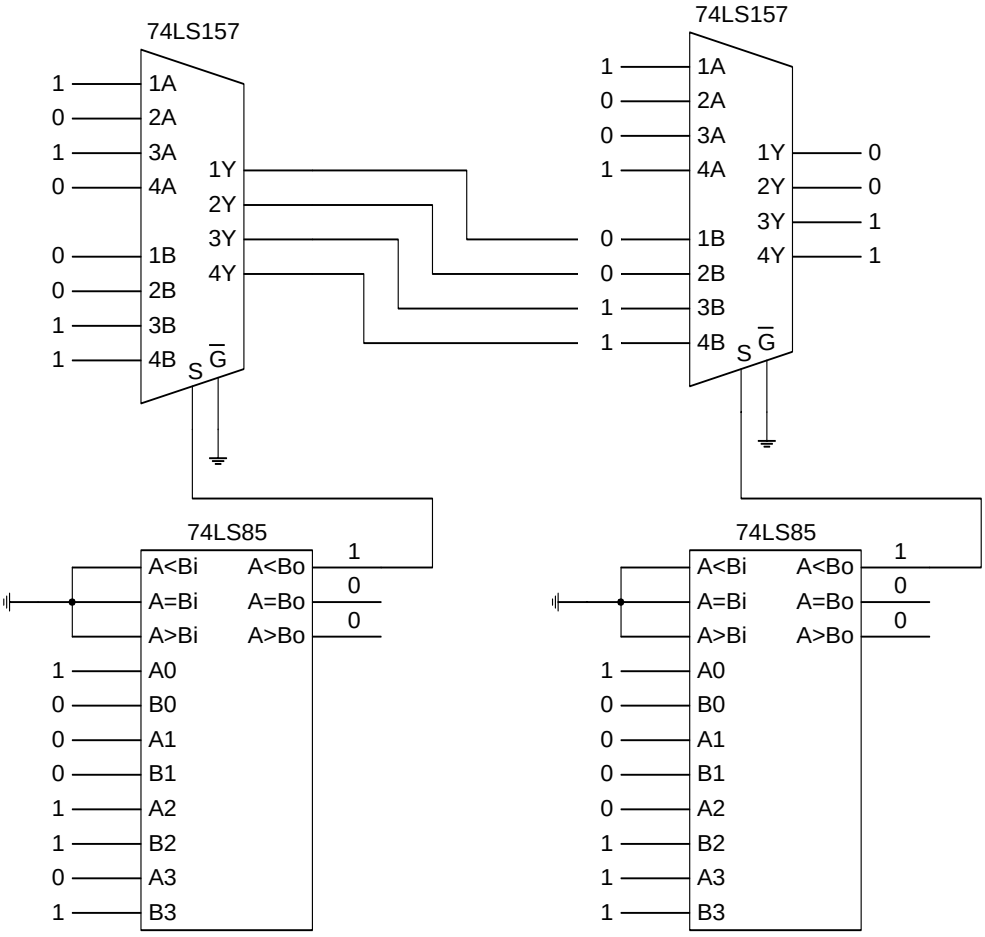
Neka se odabrani broj označi sa S. Na identičan način kao u prethodno opisanoj proceduri se odabira veći broj od brojeva R i S, tako što se broj R dovede na ulaz A komparatora i multipleksera, a broj S na ulaz B. U tom slučaju konačnu kombinacionu mrežu za odabir najvećeg broja od četvorobitnih brojeva P, Q i R prikazuje Slika 3.55.



Slika 3.55: Logička šema za određivanje najvećeg broja od tri četvorobitna broja P, Q i R

b)

Slika 3.56 prikazuje realizovanu kombinacionu mreže sa označenim vrednostima signala za dati primer.

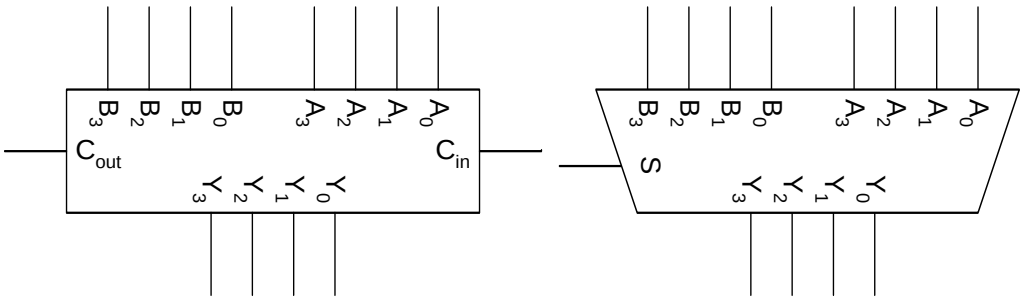


Slika 3.56: Vrednosti signala za dati primer

3.20 ZADATAK:

X i Y su četvorobitni brojevi (X=(X_3, X_2, X_1, X_0); Y=(Y_3, Y_2, Y_1, Y_0)). Ako je $X + Y \leq 15_{10}$, tada Z=(Z_3, Z_2, Z_1, Z_0) treba da bude jednako zbiru X+Y. U suprotnom slučaju, $X + Y > 15_{10}$, Z treba da bude jednako $15_{10}=1111_2$.

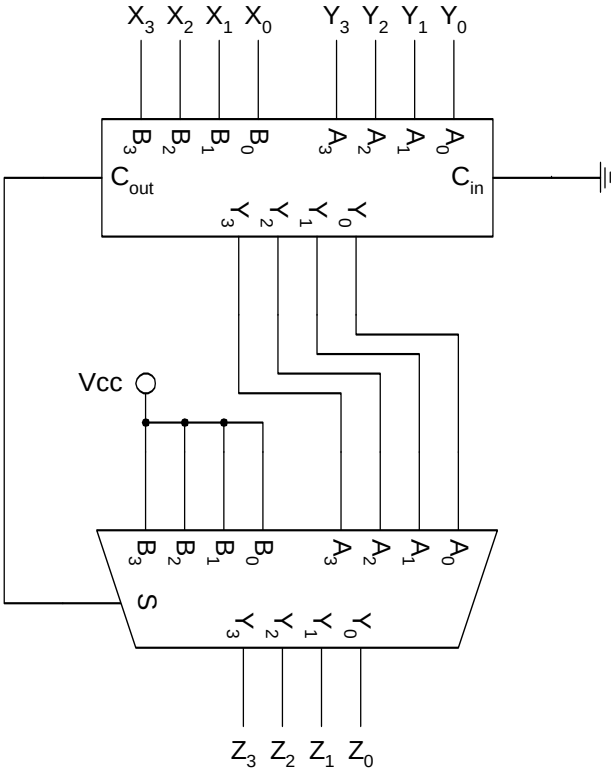
Za realizaciju stoje na raspolaganju četvorobitni sabirač i multiplekser 2×1. Njihove blok šeme prikazuje Slika 3.57.



Slika 3.57: Komponente za realizaciju traženog sabirača

REŠENJE:

Ako se na ulaze A i B sabirača dovedu brojevi X i Y, tada sabirač na izlazu Y generiše zbir X+Y pod uslovom da je ulazni prenos C_{in} jednak nuli. Zbog toga je potrebno ulaz sabirača C_{in} spojiti na masu. U sličaju da je za rezultat sabiranja potrebno više od četiri bita izlaz sabirača C_{out} postaje aktivan (C_{out}=1), tj. došlo je do pojave izlaznog prenosa. Na osnovu toga se zaključuje da u slučaju da je $X + Y > 15_{10}$ aktivan signal izlaznog prenosa.



Slika 3.58: Realizovani sabirač četvorobitnih brojeva X i Y

To se može iskoristiti za adresiranje ulaza multipleksera, pa se na ulaz multipleksera sa adresom nula dovede izlaz sabirača, dok se ulaz multipleksera sa adresom jedan postavi na $15_{10}=1111_2$. Na taj način se obezbeđuje da u slučaju kada je $X + Y \leq 15_{10}$ izlaz Z bude jednak zbiru X+Y, a u suprotnom slučaju, $X + Y > 15_{10}$, Z tada postaje $15_{10}=1111_2$.

Slika 3.58 prikazuje blok šemu povezivanja sabirača i multipleksera za realizaciju traženog sabirača.