POKAZNA VEŽBA 3

VHDL opis kombinacionih mreža

Potrebno predznanje

- Urađena pokazna vežba 2
- Standardne kombinacione mreže

Šta će biti naučeno tokom izrade vežbe?

Nakon urađene vežbe, bićete u mogućnosti da:

- Opišete proizvoljnu kombinacionu mrežu u VHDL-u koristeći opis na nivou logičkih kola
- Opišete proizvoljnu kombinacionu mrežu u VHDL-u koristeći uslovne dodele
- Opišete proizvoljnu kombinacionu mrežu u VHDL-u koristeći kombinacione procese
- Simulirate vaš sistem uz pomoć ModelSim-Altera alata

Apstrakt i motivacija

Opis digitalnih sistema na nivou logičkih kola, bilo da ga radite crtajući električnu šemu ili opisujući VHDL jezikom na nivou logičkih kola, zahteva da poznajete Bulove jednačine sistema koji želite da opišete. Svakako, opis digitalnih sistema u VHDL-u olakšava i ubrzava proces projektovanja digitalnih sistema, a u narednim vežbama ćete naučiti da razlikujete i opišete različite kombinacione i sekvencijalne sisteme. U prethodnoj vežbi smo pokazali više načina za opisivanje digitalnih sistema u VHDL-u i već tada je bilo očevidno da nije svaki opis pogodan za svaku vrstu digitalnog sistema. U ovoj vežbi ćete proširiti znanje o kombinacionim mrežama i na kraju vežbe opisati složeni kombinacioni sistem koji veoma liči na aritmetičko-logičku jedinicu nekog procesora.

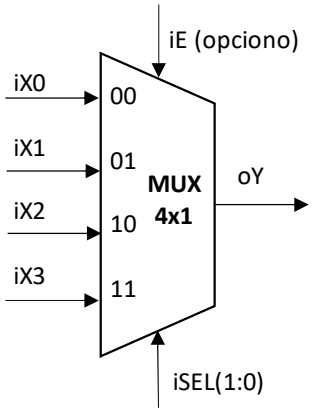
TEORIJSKE OSNOVE

1. Kombinacione mreže

Kombinacione mreže su složeni digitalni sistemi kod kojih logičko stanje izlaza zavisi samo od **trenutnih vrednosti** signala na ulazu. Osnovne vrste kombinacionih mreža su:

- Logička kola
- Multiplekser
- Demultiplekser
- Dekoder
- Koder, itd.

Multiplekser je kombinaciona mreža sa $\mathbf{2^n}$ ulaza, **n** upravljačkih (selekcionih) signala i **jednim** izlazom, gde upravljački signali određuju koji od ulaza će biti prosleđen na izlaz u datom trenutku. U nastavku je prikazan multiplekser 4x1, istinitosna tablica i primer VHDL opisa pomoću kombinacionog procesa sa if-else strukturom. **Primena**: Prenos podataka sa više izvora preko jedne linije (npr. telekomunikacioni sistemi).

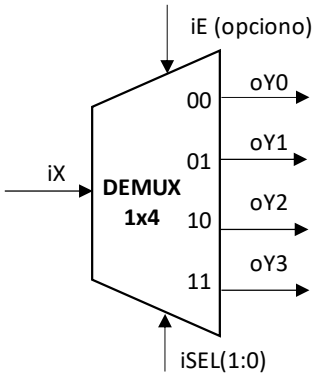


MUX 4x1			
iE	**$iSEL_1$**	**$iSEL_0$**	**oY**
0	**X**	**X**	**0**
1	**0**	**0**	**iX_0**
1	**0**	**1**	**iX_1**
1	**1**	**0**	**iX_2**
1	**1**	**1**	**iX_3**

Slika 1-1. Primer multipleksera 4x1

```
process (iE, iX0, iX1, iX2, iX3, iSEL)
begin
      if(iE = '1') then
            if (iSEL = "00") then
                  oY <= iX0;
            elsif (iSEL = "01") then
                  oY <= iX1;
            elsif (iSEL = "10") then
                  oY <= iX2;
            else
                  oY <= iX3;
            end if;
      else
            oY <= '0';
      end if;
end process;
```

Demultiplekser je kombinaciona mreža sa **jednim** ulazom, **n** upravljačkih (selekcionih) signala i $\mathbf{2^n}$ izlaza, gde se vrednost ulaznog signala prenosi na jedan od više izlaznih signala na osnovu upravljačkih (selekcionih) ulaza. U nastavku je prikazan demultiplekser 1x4, istinitosna tablica i primer VHDL opisa uslovnom dodelom.
Primena: Razdvaja podatke sa jednog ulaza na više izlaza (npr. digitalni video/audio sistemi).

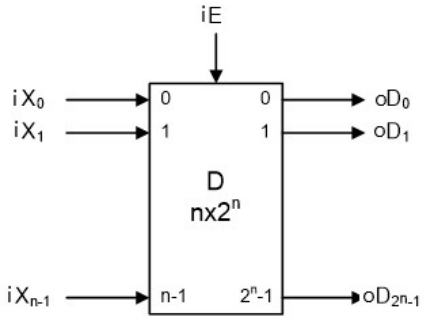


DEMUX 1x4						
iE	**$iSEL_1$**	**$iSEL_0$**	**oY_3**	**oY_2**	**oY_1**	**oY_0**
0	**X**	**X**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**0**	**iX**
1	**0**	**1**	**0**	**0**	**iX**	**0**
1	**1**	**0**	**0**	**iX**	**0**	**0**
1	**1**	**1**	**iX**	**0**	**0**	**0**

Slika 1-2. Primer demultipleksera sa 2^n izlaza i istinitosna tablica za DEMUX 1x4

```
oY0 <= iX when (iSEL = "00" and iE = '1') else '0';
oY1 <= iX when (iSEL = "01" and iE = '1') else '0';
oY2 <= iX when (iSEL = "10" and iE = '1') else '0';
oY3 <= iX when (iSEL = "11" and iE = '1') else '0';
```

Dekoder je kombinaciona mreža sa **n** ulaza i $\mathbf{2^n}$ izlaza, koja dekoduje binarni signal na ulazu i aktivira tačno jedan od svojih izlaza na osnovu vrednosti binarnog ulaza. U nastavku je prikazana istinitosna tablica dekodera 2x4, kao i primer VHDL opisa uslovnom dodelom. **Primena**: adresiranje u memorijskim uređajima, 7-segmentni displej, dekodiranje instrukcija u CPU, itd.

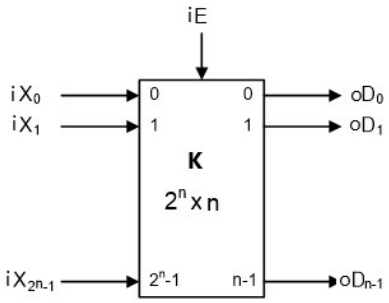


DEKODER 2x4						
iE	**iX_1**	**iX_0**	**oD_3**	**oD_2**	**oD_1**	**oD_0**
0	**X**	**X**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**0**	**1**
1	**0**	**1**	**0**	**0**	**1**	**0**
1	**1**	**0**	**0**	**1**	**0**	**0**
1	**1**	**1**	**1**	**0**	**0**	**0**

Slika 1-3. Primer dekodera sa n ulaza i 2^n izlaza i istinitosna tablica za DEKODER 2x4

```
oD3 <= '1' when (iX1 = '1' and iX0 = '1' and iE = '1') else '0';
oD2 <= '1' when (iX1 = '1' and iX0 = '0' and iE = '1') else '0';
oD1 <= '1' when (iX1 = '0' and iX0 = '1' and iE = '1') else '0';
oD0 <= '1' when (iX1 = '0' and iX0 = '0' and iE = '1') else '0';

                          ili

oD <= "0001" when (iX = "00" and iE = '1') else
      "0010" when (iX = "01" and iE = '1') else
      "0100" when (iX = "10" and iE = '1') else
      "1000" when (iX = "11" and iE = '1') else
      "0000";
```

Koder je kombinaciona mreža sa 2^n ulaza i **n** izlaza, koja na osnovu aktivnog ulaza daje binarnu reprezentaciju na izlazu. U nastavku je prikazan koder 4x2, istinitosna tablica i VHDL opis pomoću IF-ELSE strukture.
Primena: numeričke tastature, kompresija podataka, itd.



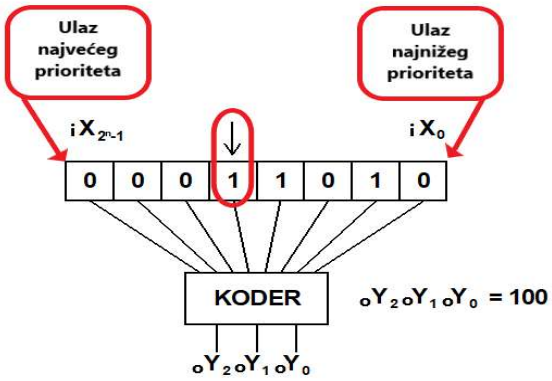
KODER 4x2						
iE	iX_3	iX_2	iX_1	iX_0	oD_1	oD_0
0	X	X	X	X	0	0
1	0	0	0	1	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	1	0	0	0	1	1

Slika 1-4. Primer kodera sa 2^n ulaza i n izlaza i istinitosna tablica za KODER 4x2

```
process (iX) begin
      if(iE = '1') then
            if (iX = „0001“) then
                  oY <= "00";
            elsif (iX = „0010“) then
                  oY <= "01";
            elsif (iX = „0100“) then
                  oY <= "10";
            else
                  oY <= "11";
            end if;
      else
            oY <= "00";
      end if;
end process;
```

Kod kodera se može javiti **PROBLEM** definisanja izlaza u slučaju kada je istovremeno aktivno više od jednog ulaza. U ovom slučaju se definiše lista prioriteta za ulazne signale i takav koder se naziva **prioritetni koder**.

U slučaju kada je više od jednog ulaza aktivno, običan koder će izbaciti pogrešnu vrednost, dok će na izlazu prioritetnog kodera biti kôd ulaza **višeg** ili **nižeg** prioriteta. Na Slici 1-5 prikazan je prioritetni koder 8x3 višeg prioriteta, koji ima više aktivnih ulaza (iX_1, iX_3 i iX_4). Obzirom da je iX_4 ulaz najvećeg prioriteta, na izlazu će biti prikazan njegov kôd $oY_2oY_1oY_0$ = 100.



Slika 1-5. Primer kodera i problema sa više aktivnih ulaza

2. Aritmetičke kombinacione mreže

Aritmetičke kombinacione mreže uglavnom izvršavaju jednu aritmetičku operaciju i tu spadaju:

- sabirač,
- komplementer (za dobijanje I i II komplementa),
- komparator,
- pomerač,
- množač, delitelj, itd.

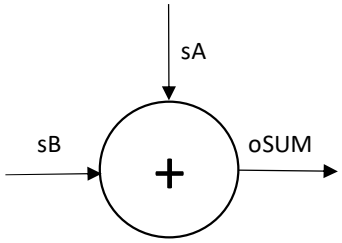
Ukoliko kombinaciona mreža izvršava više od jedne operacije naziva se **aritmetičko logička jedinica**.

2.1. Sabirač

Sabirač je kombinaciona mreža koja vrši aritmetičko sabiranje brojeva. Sabiranjem dva n-bitna broja dobija se rezultat koji ima n+1 bit. Kako bi dobili ispravan rezultat, neoznačeni sabirci se proširuju **NULOM**, dok se označeni sabirci proširuju **ZNAKOM**. Kako bismo omogućili rad sa aritmetičkim operacijama i označenim/neoznačenim operandima, potrebno je uključiti JEDNU (odgovarajuću) od naredne dve biblioteke:

```
use ieee.std_logic_unsigned.all;        -- neoznačeni brojevi
                              ili
use ieee.std_logic_signed.all;          -- označeni brojevi
```

Sledi primer sabirača u VHDL-u koji sabira osmobitne signale A i B i rezultat smešta u devetobitne signale SUM_N za neoznačen rezultat, odnosno SUM_O za označen:




```
signal sA, sB             : std_logic_vector(7 downto 0);
signal sSUM_N, sSUM_O     : std_logic_vector(8 downto 0);

sSUM_N <= ('0' & sA) + ('0' & sB);        -- neoznačeni brojevi
sSUM_O <= (sA(7) & sA) + (sB(7) & sB);    -- označeni brojevi
```

VAŽNA NAPOMENA: Ukoliko rezultat u sistemu **ima isti broj bita kao operandi**, prenos se ignoriše! U tom slučaju biste prilikom sabiranja četvorobitnih vrednosti 9 i 7 (1001 + 0111) dobili rezultat 0000 jer se jedinica na najvišoj poziciji ignoriše i u tom slučaju nije potrebno proširivati sabirke.

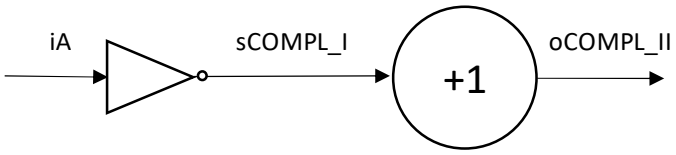
```
signal sA, sB             : std_logic_vector(7 downto 0);
signal sSUM_N, sSUM_O     : std_logic_vector(7 downto 0);

sSUM_N <= sA + sB;        -- neoznačeni brojevi
sSUM_O <= sA + sB;        -- označeni brojevi
```

2.2. Komplementer

Komplement je dopuna datog broja do neke unapred definisane vrednosti. Najčešće se koristi za prikazivanje negativnih brojeva ili za realizaciju oduzimanja pomoću sabiranja. U binarnom brojnom sistemu (osnove 2), mogu da se definišu samo dva komplementa:

- **Komplement jedinice (I komplement) –** dobijamo negiranjem svakog bita polaznog binarnog broja
- **Komplement dvojke (II komplement) –** dobijamo dodavanjem jedinice na kompl. I zadatog broja

Komplementer je kombinaciona mreža koja računa komplement I ili II, za zadatu vrednost ulaznog signala.



U **decimalnom** brojnom sistemu, pozitivne i negativne brojeve razlikujemo po znaku ispred apsolutne vrednosti broja (+ ili -).

Ukoliko u **binarnom** brojnom sistemu koristimo predstavu broja u formatu **DIREKTNOG KODA (ZNAK + APSOLUTNA VREDNOST)**, tada najznačajniji bit MSB definiše znak, a ostali biti sadrže apsolutnu vrednost broja. Ukoliko je MSB = 1, broj je negativan, dok je za MSB = 0 broj pozitivan. Problem kod ovog načina predstavljanja je što se ne dobija tačan rezultat ako u sabiranju učestvuju negativni brojevi. Jedan od najčešće korišćenih načina predstavljanja negativnih binarnih brojeva, kojim se rešava problem sa aritmetičkim operacijama nad negativnim brojevima, je **komplement dvojke**.

U narednom primeru ćemo predstaviti broj $-7_{(10)}$ u komplementu dvojke:

$7_{(10)} = 0111_{(2)}$

Prvi komplement => $1000_{(2)}$

Drugi komplement => $1000_{(2)} + 1 = 1001_{(2)}$

$-7_{(10)} = 1001_{(2)}$

Provera:

$+7_{(10)} = 0111_{(2)} = 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 4 + 2 + 1 = 7$

$-7_{(10)} = 1001_{(2)} = -1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = -8 + 1 = -7$

U nastavku su u Tabeli 2-1 dati primeri sabiranja brojeva zapisanih u komplementu dvojke:

Tabela 2-1. Sabiranje brojeva zapisanih u komplementu dvojke

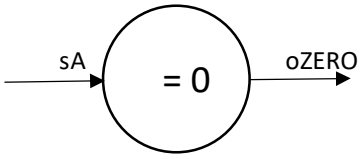
A	+2	0010	+2	0010	-5	1011	+6	0110
B	+3	0011	-7	1001	-2	1110	+7	0111
A + B	+5	0101	-5	1011	-7	11001	+13	1101 = -3

Jedini ispravan način predstavljanja negativnog broja u računaru jeste pomoću komplement II predstave.

```
sCOMPL_I <= not(sA);                -- komplement I
oCOMPL_II <= sCOMPL_I + 1;          -- komplement II
```

2.3. Komparator

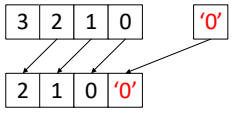
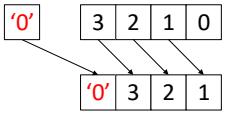
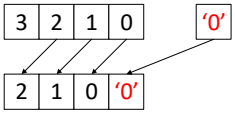
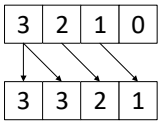
Komparator je kombinaciona mreža koja vrši poređenje vrednosti ulaznih signala. U prethodnoj vežbi smo već upoznali ovu kombinacionu mrežu i opisali je na više načina u VHDL-u. Svakako je najpogodniji način opisivanja pomoću uslovne dodele. U nastavku je dat primer poređenja signala A sa nulom:




```
oZERO <= '1' when sA = 0 else '0';
```

2.4. Pomerač

Pomerač je kombinaciona mreža koja vrši logičko ili aritmetičko pomeranje vrednosti ulaznih signala. Kod logičkog pomeranja se uvek dodaju **NULE** sa leve ili desne strane u zavisnosti od pomeranja, dok se kod aritmetičkog pomeranja udesno zadržava **ZNAK**.

Vrsta	Smer	Opis	Slika	Primer u VHDL-u
Logičko	ULEVO	**Množenje neoznačenih brojeva sa 2^n, gde je n broj pomeranja.**		Logičko pomeranje signala A ulevo za 3 mesta: `sRES <= sA(4 downto 0) & „000“;`
	UDESNO	**Deljenje neoznačenih brojeva sa 2^n, gde je n broj pomeranja**		Logičko pomeranje signala A udesno za 4 mesta: `sRES <= „0000“ & sA(7 downto 4);`
Aritmetičko	ULEVO	**Množenje označenih brojeva sa 2^n, gde je n broj pomeranja**		Aritmetičko pomeranje signala A ulevo za 1 mesto: `sRES <= sA(6 downto 0) & '0';`
	UDESNO	**Deljenje označenih brojeva sa 2^n, gde je n broj pomeranja**		Aritmetičko pomeranje signala A udesno za 2 mesta: `sRES <= sA(7)& sA(7)& sA(7 downto 2);`

NAPOMENA: Postoji više različitih teorija za realizaciju aritmetičkog pomeranja ulevo i sve su ispravne. Na vežbama ćemo koristiti teoriju da se aritmetičko pomeranje ulevo ponaša isto kao i logičko pomeranje ulevo.

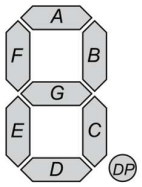
PITANJE: Da li pomerač može da pomera **svoju vrednost** tj. da napišemo u VHDL-u sledeće:

```
sRES <= sRES(5 downto 0) & „00“; ?
```

Odgovor je NE! Svaka kombinaciona mreža zavisi samo od trenutnih vrednosti na ulazu i povratna sprega u kombinacionim sistemima ne postoji! Ovaj „problem“ će nam rešiti upravo sekvencijalne mreže, koje ćemo upoznati na narednim vežbama.

2.5. Dekoder – primena na 7-segmentni displej

Dekoder za 7-segmentni displej je kombinaciona mreža koja na osnovu binarnog ulaza (najčešće 4-bitni binarni broj) generiše odgovarajuće izlaze koji kontrolišu segmente 7-segmentnog displeja, kako bi se prikazala cifra (0-9) ili slovo. Na Slici 2-5 je prikazan 7-segmentni displej, koji se sastoji od 7 segmenata (3 horizontalna i 4 uspravna) gde je svaki segment zapravo nezavisna LED dioda: **a, b, c, d, e, f, g**. Za prikaz decimalne tačke koristi se poseban segment **dp**. Platforma MAX 1000 ima četiri 7-segmentna displeja.



Slika 2-5. Izgled 7-segmentnog displeja

Brojevi od 0 do 9, kao i slova ili heksadecimalne cifre A, B, C, D, E i F, mogu se ispisati aktiviranjem odgovarajućih delova 7-segmentnog displeja, kao što je prikazano na Slici 2-6. Svaki od segmenata se aktivira tj. svetli podešavanjem odgovarajućeg pina. Listu sa svim pinovima možete pronaći u dokumentu *LPRS1_FPGA_pins.pdf*.

Slika 2-6. Ispis brojeva i slova na 7-segmentnom displeju

U Tabeli 1-1 data je istinitosna tablica u kojoj se vidi koje segmente je potrebno aktivirati kako bi bio prikazan svaki od brojeva 0 do 9.

NAPOMENA: **Segmenti 7-segmentnog displeja na MAX 10 platformi su aktivni na 0, tako da ih treba invertovati u odnosu na poslednju kolonu u Tabeli 1-1.**

Tabela 1-1. Istinitosna tablica za prikaz brojeva 0-9 na 7-segmentni displej

Dekadna cifra (hexadecimalna)	Binarni format				Segmenti na 7-segmentnom displeju						
	B3	B2	B1	B0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

ZADACI

1. Multiplekser

Vaš zadatak je da implementirate 4x1 multiplekser bez signala dozvole (4 ulaza opšte namene, 2 selekciona ulaza i 1 izlaz) na sledeće načine:

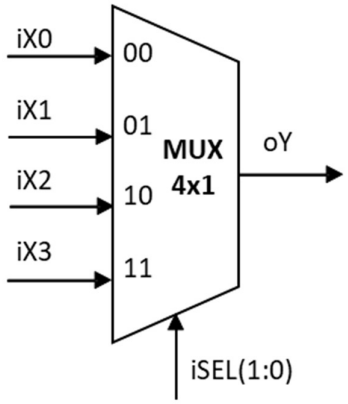
- opisom u VHDL-u na nivou logičkih kola,
- opisom u VHDL-u koristeći uslovnu dodelu,
- opisom u VHDL-u koristeći kombinacioni proces za IF-ELSE,
- opisom u VHDL-u koristeći kombinacioni proces za CASE.

Nakon opisa sistema u VHDL-u, proverite ispravnost vašeg sistema u ModelSim-Altera simulatoru.

Prvi korak u implementaciji sistema je analiza broja prolaza i pravljenje istinitosne tablice. Multiplekser 4x1 ima sledeće prolaze:

- 4 ulaza opšte namene – iX3, iX2, iX1, iX0
- 2 selekciona ulaza koji određuju koji ulaz će se prosleđivati na izlaz – iSEL1 i iSEL0
- 1 izlaz – oY

Istinitosna tablica multipleksera 4x1 je data u nastavku:



MUX 4x1		
$iSEL_1$	$iSEL_0$	oY
0	0	iX_0
0	1	iX_1
1	0	iX_2
1	1	iX_3

Na osnovu istinitosne tablice možemo izvesti logičku funkciju koja će nam služiti za opis sistema u VHDL-u na nivou logičkih kola:

$$Y = iX_0\,\overline{iSEL_1}\,\overline{iSEL_0} + iX_1\,\overline{iSEL_1}\,iSEL_0 + iX_2\,iSEL_1\,\overline{iSEL_0} + iX_3\,iSEL_1\,iSEL_0$$

S obzirom na to da jednom izlaznom signalu ne možemo dodeliti više vrednosti i da ovaj zadatak treba uraditi na 4 načina, u entitet ćemo dodati posebne izlaze za svaki traženi opis. Na taj način ćemo biti u mogućnosti da u simulaciji proverimo da li su sva rešenja identična u istom trenutku za istu kombinaciju ulaza.

Dva selekciona signala smo spojili u jedan dvobitan signal iSEL korišćenjem tipa ***std_logic_vector.***

U nastavku je dat opis ovog digitalnog sistema u VHDL-u:

```
library ieee;
use ieee.std_logic_1164.all;
entity mux is
   port(
         iX0   : in std_logic;
         iX1   : in std_logic;
         iX2   : in std_logic;
         iX3   : in std_logic;
         iSEL  : in std_logic_vector(1 downto 0);
         oY_logicka_kola   : out std_logic;
         oY_uslovna_dodela : out std_logic;
         oY_process_IF     : out std_logic;
         oY_process_CASE   : out std_logic
       );
end entity;

architecture Behavioral of mux is
   signal sS0, sS1, sS2, sS3 : std_logic;
begin
   -- logicka kola
   sS0 <= iX0 and not(iSEL(1)) and not(iSEL(0));
   sS1 <= iX1 and not(iSEL(1)) and iSEL(0);
   sS2 <= iX2 and iSEL(1) and not(iSEL(0));
   sS3 <= iX3 and iSEL(1) and iSEL(0);
   oY_logicka_kola <= sS0 or sS1 or sS2 or sS3;

   -- uslovna dodela
   oY_uslovna_dodela   <=  iX0 when iSEL = "00" else
                           iX1 when iSEL = "01" else
                           iX2 when iSEL = "10" else
                           iX3;
   -- if-else
   process (iX0, iX1, iX2, iX3, iSEL) begin
         if (iSEL = "00") then
               oY_process_IF <= iX0;
         elsif (iSEL = "01") then
               oY_process_IF <= iX1;
         elsif (iSEL = "10") then
               oY_process_IF <= iX2;
         else
               oY_process_IF <= iX3;
         end if;
   end process;

   -- case
   process (iX0, iX1, iX2, iX3, iSEL) begin
         case (iSEL) is
               when "00"   => oY_process_CASE <= iX0;
               when "01"   => oY_process_CASE <= iX1;
               when "10"   => oY_process_CASE <= iX2;
               when others => oY_process_CASE <= iX3;
         end case;
   end process;
end architecture;
```

NAPOMENA: U svakom od prethodnih opisa multipleksera možete primetiti da je obavezna podrazumevana grana (**ELSE** ili **WHEN OTHERS**) i da se u njoj dodeljuje jedna od postojećih ulaza multipleksera.

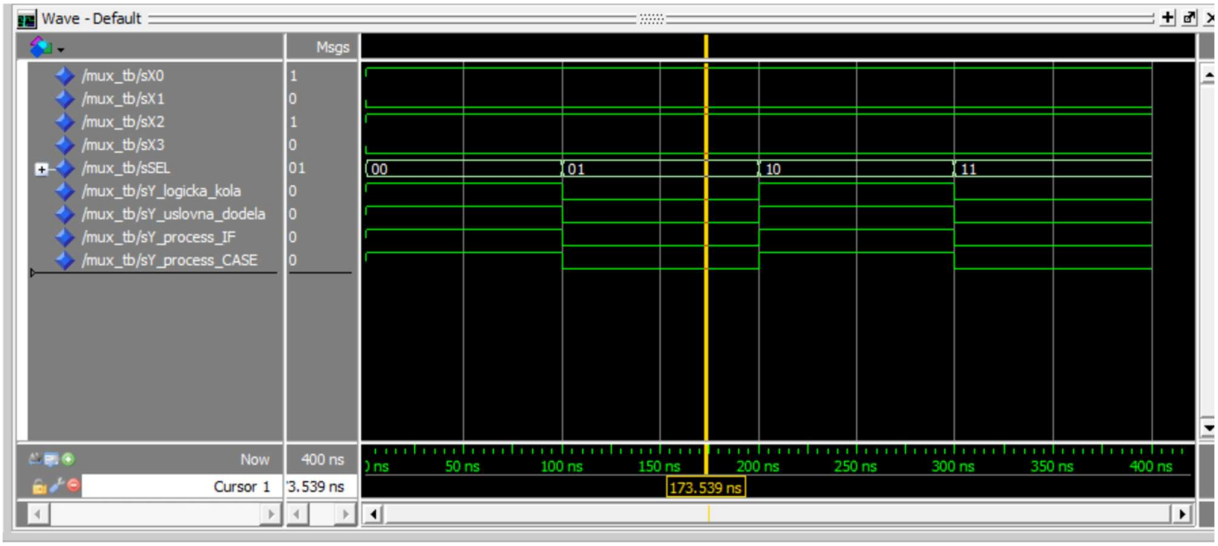
U nastavku sledi primer kako **NIJE ISPRAVNO** opisati multiplekser ili bilo koji digitalni sistem sa konačnim brojem ulaza/izlaza. Konkretno, multiplekser u ovom primeru ima samo 4 ulaza i stoga ne može biti više od 4 CASE grane koje dodeljuju vrednosti izlaznom signalu:

```
process (iX0, iX1, iX2, iX3, iSEL) begin
        case (iSEL) is
              when "00"   => oY_process_CASE <= iX0;
              when "01"   => oY_process_CASE <= iX1;
              when "10"   => oY_process_CASE <= iX2;
              when "11"   => oY_process_CASE <= iX3; -- NIJE ISPRAVNO!
              when others => oY_process_CASE <= iX3;
        end case;
   end process;
```

Sledeći korak je pisanje test bench datoteke, kako bi se sistem simulirao u ModelSim-Altera alatu. U stimulus procesu je potrebno postaviti vrednosti ulaza iX3, iX2, iX1 i iX0 na bilo koje vrednosti, a zatim zadati sve kombinacije vrednosti iSEL signala iz istinitosne tablice. U nastavku je dat VHDL opis stimulus procesa u test bench-u:

```
stimulus: process
   begin
         sX0 <= '1';
         sX1 <= '0';
         sX2 <= '1';
         sX3 <= '0';

         sSEL <= "00";
         wait for 100 ns;

         sSEL <= "01";
         wait for 100 ns;

         sSEL <= "10";
         wait for 100 ns;

         sSEL <= "11";
         wait;

   end process stimulus;
```

Na slici 1-1 sledi prikaz rezultata u simulatoru, gde smo potvrdili ispravnost sistema za sva četiri načina opisivanja sistema:



Slika 1-1. Rezultat simulacije za MUX 4x1

Nakon potvrđene ispravnosti sistema u simulatoru, proverite ispravnost na platformi MAX 10. U nastavku sledi predlog za povezivanje ulaza i izlaza na odgovarajuće pinove:

Prolazi	Smer	Komponenta	FPGA pin	I/O standard
iX3	input	I_SW[3]	PIN_M1	3.3 V Schmitt Trigger
iX2	input	I_SW[2]	PIN_M2	3.3 V Schmitt Trigger
iX1	input	I_SW[1]	PIN_L3	3.3 V Schmitt Trigger
iX0	input	I_SW[0]	PIN_M3	3.3 V Schmitt Trigger
iSEL1	input	I_PB_LEFT	PIN_D1	3.3 V Schmitt Trigger
iSEL0	input	I_PB_RIGHT	PIN_E3	3.3 V Schmitt Trigger
oY	output	o_sem_g	PIN_B1	3.3-V LVTTL

2. Demultiplekser (dodatni zadatak)

Implementirati 1x4 demultiplekser bez signala dozvole (1 ulaz opšte namene, 2 selekciona ulaza i 4 izlaza) na sledeće načine:

- opisom u VHDL-u na nivou logičkih kola,
- opisom u VHDL-u koristeći uslovnu dodelu,
- opisom u VHDL-u koristeći kombinacioni proces.

Nakon opisa sistema u VHDL-u, proverite ispravnost vašeg sistema simulacijom.

3. Prioritetni koder (dodatni zadatak)

Implementirati prioritetni koder bez signala dozvole sa 8 ulaza na sledeće načine:

- opisom u VHDL-u na nivou logičkih kola,
- opisom u VHDL-u koristeći uslovnu dodelu,
- opisom u VHDL-u koristeći kombinacioni proces.

Nakon opisa sistema u VHDL-u, proverite ispravnost vašeg sistema simulacijom.

4. Složeni kombinacioni sistem 1

U VHDL jeziku opisati i simulirati digitalni sistem čije je ponašanje opisano u nastavku:

Ulazi digitalnog sistema:

- **iA [7:0]** – prvi ulazni operand,
- **iB [2:0]** – drugi ulazni operand,
- **iSEL** – izbor operacije (sami odredite koliko bita treba da ima signal).

Izlazi digitalnog sistema:

- **oY** – rezultat operacije (sami odredite koliko bita treba da ima signal),
- **oSEGM[6:0]** – izlazni signal za 7-segmentni displej
- **oZERO** – izlazni signal koji pokazuje da li je rezultat operacije jednak nuli.

Opis digitalnog sistema:

Na signal ***sENC*** dovesti indeks bita na logičkoj jedinici signala ***iA***. Ukoliko signal ***iA*** ima više bita koji sadrže jedinicu, izabrati onaj sa najvišim indeksom. Ako nijedan bit nije na logičkoj jedinici, neka rezultat bude jednak 3.

Signalu ***sADD*** dodeliti zbir neoznačenih vrednosti signala ***sENC*** i ***iB.*** Signal ***sADD*** treba da sadrži i prenos prilikom sabiranja.

Signalu ***sCOMPL*** dodeliti suprotnu vrednost signala ***sADD***. Ako je na ulazu X, komponenta treba da odredi –X.

Na signal ***sSHR*** treba dovesti signal ***sADD*** koji je aritmetički pomeren za 2 bita udesno.

Na signalu ***sDEC*** postaviti na logičku jedinicu bit sa indeksom ***iB***, a sve ostale bite na logičku NULU.

Signalu ***sMUX*** dodeliti :

- ***sCOMPL*** ako je selekcioni signal ***iSEL*** jednak 0
- ***sSHR*** ako je selekcioni signal ***iSEL*** jednak 1
- ***sDEC[7:4]*** ukoliko je selekcioni signal ***iSEL*** jednak 2
- ***sDEC[3:0]*** ukoliko je selekcioni signal ***iSEL*** jednak 3

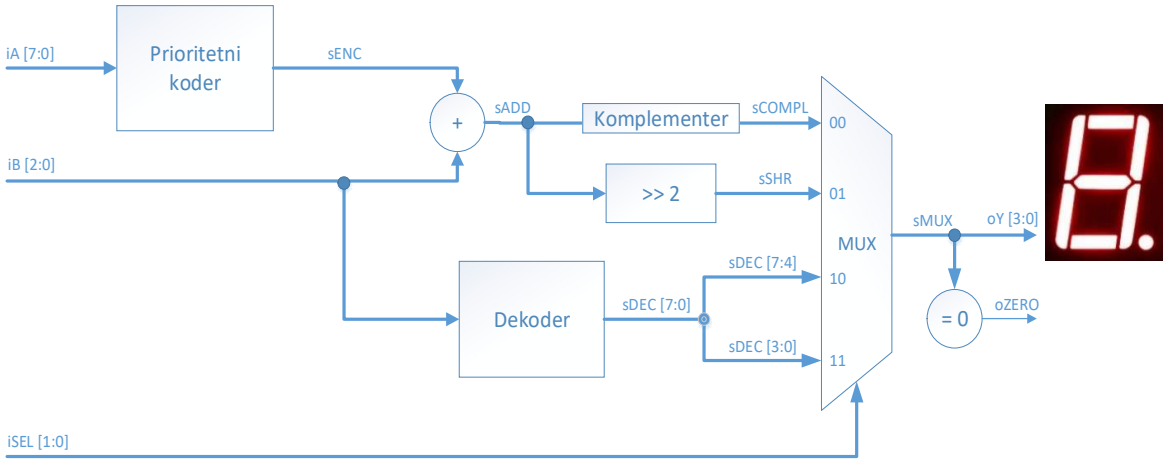
Izlaznom signalu ***oY*** dodeliti signal ***sMUX.***

Signal ***sMUX*** dekodovati u signal ***oSEGM[6:0]*** i prikazati na **7-segmentnom displeju.**

Izlaznom signalu ***oZERO*** dodeliti logičku jedinicu ukoliko je ***sMUX = 0.***

Nacrtati blok dijagram šemu i opisati digitalni sistem u VHDL-u. Sistem simulirati izvršavajući svaku od operacija bar jednom, sa bar 2 različite vrednosti na svakom operandu. Bar jedan od slučajeva treba da rezultuje nulom i bar jedan od slučajeva brojem različitim od nule.

REŠENJE:



Ovaj digitalni sistem predstavlja jednostavnu aritmetičko-logičku jedinicu koja podržava sledeće operacije:

- zbir dva neoznačena ulazna operanda,
- negacija ulaznog operanda (od iS izračunati vrednost –iS), odnosno komplement II ulazne vrednosti,
- zbir dva ulazna operanda podeljen sa 4, tj. aritmetički pomeren u desno za 2 mesta,
- dekodiranje ulazne vrednosti iB i dodelu gornja ili donja 4 bita dekodovane vrednosti.

Izlaz iz multipleksera se prosleđuje na izlaz oY, kao i komparatoru koji poredi rezultat sa NULOM. Ukoliko je izlaz iz multipleksera jednak NULI, izlazni signal oZERO će biti aktivan tj. jednak jedinici (oZERO = '1').

PITANJE: Da li izlazni signal oY može da bude ulaz u komparator?

5. Složeni kombinacioni sistem 2

Vaš naredni zadatak je da implementirate složeniji digitalni sistem, sa dva 4-bitna ulaza (iA i iB), jednim selekcionim ulazom od 2 bita (iSEL) i jednim 4-bitnim izlazom (oC), tako da se sistem ponaša prema sledećoj jednačini:

$$oC = \begin{cases} iA & if\ iSEL = "00" \\ iB & if\ iSEL = "01" \\ iA + \boldsymbol{sB} & if\ iSEL = "10" \\ \boldsymbol{sB} + 1 & if\ iSEL = "11" \end{cases}$$

Ulaz iB je u formatu direktnog koda tj. **ZNAK + MODUO od 3 bita** i potrebno ga je konvertovati u format **drugog komplementa**, kako bi se mogle izvršiti zadate aritmetičke operacije.

$$iB\ je\ u\ formatu\ \boldsymbol{ZNAK + MODUO}: \quad iB(3)\ \&\ iB(2\ \boldsymbol{downto}\ 0)$$

$$\boldsymbol{sB} = \begin{cases} not\big('0'\ \&\ iB(2\ \boldsymbol{downto}\ 0)\big) + 1 & if\ iB < 0 \\ iB & else \end{cases}$$

NAPOMENA: Kako bi se pravilno koristila operacija sabiranja, potrebno je na početku vhdl datoteke uključiti **STD_LOGIC_UNSIGNED** biblioteku, za korišćenje neoznačenih brojeva.

```
use ieee.std_logic_unsigned.all;
```

Primetite da ovaj sistem sadrži 10 ulaza i 4 izlaza, što znači da on računa 4 Bulove funkcije od 10 promenljivih. Tradicionalni pristup projektovanju digitalnih sistema kroz istinitosne tablice i crtanjem logičke šeme bi trajao veoma dugo, a VHDL nam omogućuje opis istog sistema u samo nekoliko linija! **Ali nikada ne treba zaboraviti da ono što mi opisujemo VHDL-om jesu logička kola i jesu Bulove funkcije izvedene iz istinitosne tablice! VHDL je samo drugi (jednostavniji) način da se opiše isti sistem.**

PITANJE: Koja kombinaciona mreža je opisana u ovom digitalnog sistemu?

ZAKLJUČAK

Koristeći uslovne dodele i kombinacione procese moguće je opisati veoma složenu Bulovu funkciju na kraći i razumljiviji način. Nakon završene laboratorijske vežbe, trebali biste biti u stanju da opišete proizvoljno složenu kombinacionu mrežu koristeći VHDL konstrukcije i tradicionalno, pomoću Bulovih funkcija.