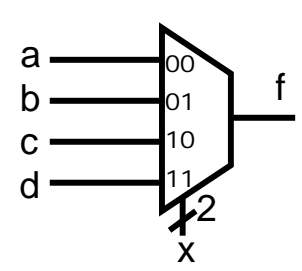
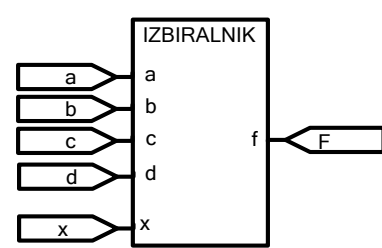


## RAZVOJ DIGITALNIH SISTEMOV

### VAJA 2: ŠTIRIMESTNI 7-SEGMENTNI LED PRIKAZOVALNIK

#### IZBIRNI STAVEK

Izbirni stavek (`with select`) lahko realizira izbiralnik (multiplekser). Z izbirnim stavkom lahko enemu izhodu priredimo podatkovni vhod, ki je izbran z vrednostjo naslovnega vhoda.

<pre>library IEEE; use IEEE.STD_LOGIC_1164.ALL;  entity izbiralnik is     Port ( x : in STD_LOGIC_VECTOR(1 downto 0);           a, b, c, d : in STD_LOGIC;           f : out STD_LOGIC); end izbiralnik;  architecture arch of izbiralnik is begin with x select     f &lt;= a when "00",          b when "01",          c when "10",          d when "11"; end arch;</pre>	<p>Predstavitev s simbolom izbiralnika:</p>  <p>VHDL predstavitev komponente:</p> 
---	--

Na zgornji sliki je prikazan primer enostavnega izbiralnika 4/1, ki z dvobitnim naslovnim signalom ( $x$ ) izbira med štirimi podatkovnimi vhodi ( $a$ ,  $b$ ,  $c$ ,  $d$ ) in to vrednost pošlje na izhod izbiralnika ( $f$ ). V primeru, da je naslovni vhod ( $x = "00"$ ) bo izbiralnik priredil izhodnemu signalu ( $f$ ) vrednost ( $a$ ). Podobno velja za naslovni vhod ( $x = "01"$ ), ko bo izbiralnik priredil izhodnemu signalu ( $f \leq b$ ) ter za naslovni vhod ( $x = "10"$ ), ko bo izbiralnik priredil izhodnemu signalu ( $f \leq c$ ). Za preostale vrednosti naslovnega vhoda ( $x = "11"$ ), za opis katere uporabimo rezervirano besedo (`others`), bo izbiralnik priredil izhodnemu signalu ( $f \leq d$ ). Opisani primer je namenjen enostavni predstavi – izbirni stavek lahko realizira splošni prekodirnik (ang. *code converter*): Ni namreč nujno, da so podatkovni vhodi samo 1-bitni in da je naslovni vhod samo 2-biten ter vhod 1-biten. Tudi pogoji primerjave niso nujno konstante, ampak so lahko VHDL izrazi, ki se ovrednotijo v vrednost. Dimenziji vektorjev vhoda in izhoda sta vedno enaki.

2.1 Ustvarite nov projekt (Vaja2\_1) in realizirajte vezje prekodirnika (bin2led.vhd), katerega arhitektura (arch) pretvori 4-bitno dvojiško vhodno vrednost (BIN) v 4-bitno izhodno vrednost (LED), ki predstavlja kodo sedmih segmentov na LED prikazovalniku. Vektorski vhod BIN(3:0) se nahaja na stikalih SW3 do SW0. Stikalo SW3 predstavlja MSB bit vektorja BIN(3). Izhodni vektor LED(6:0) služi prižiganju ustreznih segmentov posameznega LED prikazovalnika. Delovanje prekodirnika ilustrira spodnja tabela. Vrednosti prekodirne tabele kopirajte iz navodil vaje v VHDL datoteko in jo preoblikujte v izbirni stavek (with select).

Vezju dodajte še en prekodirnik, ki določa kateri LED prikazovalnik je trenutno prižgan. Ta prekodirnik prekodira vektorski vhod DIGIT(3:0) v vektorski izhod nDIGIT(3:0). Vektorski vhod DIGIT(3:0) se nahaja na stikalih SW4 do SW1. Stikalo SW4 predstavlja MSB bit vektorja DIGIT(3). Vektorski izhod prekodirnika nDIGIT(3:0) je vezan na LED prikazovalnike. MSB bit vektorja DIGIT(3) je vezan na skrajno levi prikazovalnik (mesto tisočic). Drugi prikazovalnik z leve prižgemo tako, da postavimo drugi vhod prekodirnika na '1' (DIGIT="0100"). Na izhodu prekodirnika se mora postaviti (nDIGIT="1011"). Izhod prekodirnika (nDIGIT) namreč krmili bazo PNP tranzistorja posameznega prikazovalnika, zato prikazovalnike prižigamo z negativno logiko ('0' = prižgan). Od tod izvira tudi izbira imen signalov: Vhodni signal v pozitivni logiki ima ime (DIGIT), izhodni signal v negativni logiki pa (nDIGIT).

Realizacija signala nDIGIT:

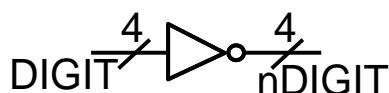
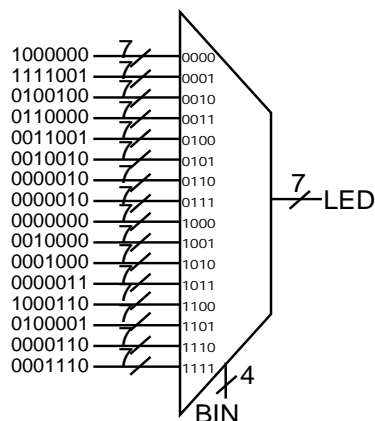


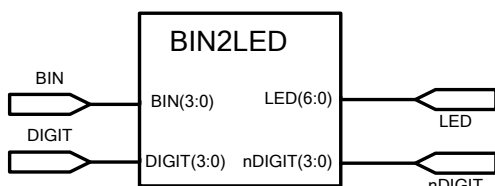
Tabela pretvorbe BIN→LED:

izhod LED	vhod BIN	opomba
1000000	0000	--0 <sub>16</sub>
1111001	0001	--1 <sub>16</sub>
0100100	0010	--2 <sub>16</sub>
0110000	0011	--3 <sub>16</sub>
0011001	0100	--4 <sub>16</sub>
0010010	0101	--5 <sub>16</sub>
0000010	0110	--6 <sub>16</sub>
1111000	0111	--7 <sub>16</sub>
0000000	1000	--8 <sub>16</sub>
0010000	1001	--9 <sub>16</sub>
0001000	1010	--A <sub>16</sub>
0000011	1011	--b <sub>16</sub>
1000110	1100	--C <sub>16</sub>
0100001	1101	--d <sub>16</sub>
0000110	1110	--E <sub>16</sub>
0001110	1111	--F <sub>16</sub>

Realizacija prekodirnika BIN→7 segmentni LED prikazovalnik:



Komponenta prekodirnika BIN→LED:



```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bin2led is
Port ( BIN, DIGIT : in STD_LOGIC_VECTOR (3 downto 0);
      LED : out STD_LOGIC_VECTOR (6 downto 0);
      nDIGIT : out STD_LOGIC_VECTOR (3 downto 0));
end bin2led;

architecture arch of bin2led is
begin
end arch;

```

#### PROCESNI STAVEK (process)

Namenjen je pisanju sekvenčno določenih izrazov, pri katerih je zaporedje ukazov pomembno (npr. pogojni stavek). Vrednosti posameznim signalom se priredijo ob koncu procesnega stavka. Ta stavek se izvede, če se spremeni katerakoli vrednost signala iz seznama občutljivosti (ang. sensitivity list).

OZNAKA: **process** (seznam\_obcutljivosti)  
**begin**  
*--sekvenčni stavki;*  
**end process;**

#### POGOJNI STAVEK (if-else)

Če je pogoj izpolnjen, se izvedejo sekvenčni stavki1, če ni sekvenčni stavki2. Pogojni stavek lahko uporabljamo samo znotraj procesnega stavka, nikakor kombinacijsko.

```

if pogoj then
    --sekvenčni stavki1;
else
    --sekvenčni stavki2;
end if;

```

#### Primer uporabe procesnega in pogojnega stavka

Prikazana je arhitektura in entiteta rotacijskega registra z vzporednim izhodom (Q), ki je prožen na sprednji rob signala ure (CLK). Register rotira vsebino za eno mesto desno na spremembo prednjega roba signala CLK.

```

entity rotreg3bit is
Port(CLK : in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR (2 downto 0)
);
end rotreg3bit;

architecture arch of rotreg3bit is
signal REG : STD_LOGIC_VECTOR(2 downto 0):="001";
begin
    P1: process(CLK)
    begin
        if (CLK'event and CLK='1') then
            REG <= REG(0) & REG(2 downto 1);
        end if;
    end process;
    Q <= REG;
end arch;

```

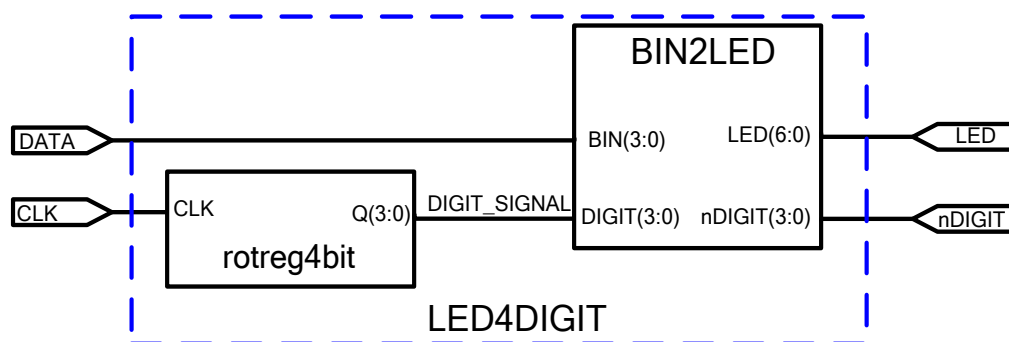
Procesni stavek P1 se izvede vedno, ko se signalu CLK spremeni vrednost (CLK' **event**) in če ima signal ob tem vrednost '1' (CLK='1'). Če pogoja združimo (**and**), se pogojni stavek izvede na vsako spremembo sprednjega roba signala CLK. Pomikanje vsebine je izvedeno z operatorjem lepljenja (&), s katerim ob vsakem sprednjem robu signala ure (CLK) novo interno vrednost signala REG sestavimo tako, da na MSB mesto postavimo trenutno LSB mesto (REG(0)), mesto nižje REG(2) in na trenutno LSB mesto (REG(1)). S tem se ob vsakem sprednjem robu signala ure (CLK) vsebina registra pomakne eno mesto desno.

- 2.2 Ustvarite nov projekt (Vaja2\_2) in realizirajte vezje rotacijskega registra (rotreg4bit.vhd), katerega arhitektura (arch) rotira 4-bitno dvojiško vsebino registra (Q) ob vsakem prednjem robu signala ure (CLK). Opazujte spreminjanje vsebine registra (Q), če na vhod za signal ure vežete tipko (BTN3). Smer rotacije ni pomembna.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rotreg4bit is
    Port ( CLK : in  STD_LOGIC;
          Q  : out STD_LOGIC_VECTOR (3 downto 0));
end rotreg4bit;
architecture arch of rotreg4bit is
begin
end arch;
```



- 2.3 Ustvarite nov projekt (Vaja2\_3), v katerem z uporabo povezovalnega stavka (port map) povežite strukturi rotacijskega registra (rotreg4bit.vhd) in prekodirnika (bin2led.vhd) v novi datoteki (led4digit.vhd), ki bo predstavljal krmilnik za 4 mestni LED prikazovalnik. Vzporedni izhod rotacijskega registra (Q) povežite na vhod prekodirnika (DIGIT). Na vhodni signal ure rotacijskega registra (CLK) povežite vhodni signal ure 4 mestnega LED prikazovalnika (CLK). Vektorski vhod BIN(3:0) se nahaja na stikalih SW0 do SW3. Stikalo SW3 predstavlja MSB bit vektorja BIN(3). Vektorski izhod (LED) povežite soležno po bitih na izhodni signal (LED) 4 mestnega LED prikazovalnika. Vektorski izhod prekodirnika (nDIGIT) povežite na izhodni signal (nDIGIT) 4 mestnega LED prikazovalnika preko internega signala (DIGIT\_SIGNAL). V vektorskem signalu (nDIGIT) je MSB bit vezan na skrajno levi prikazovalnik (tisočice). Vhod signala ure pomikalnega registra (CLK) v UCF datoteki je na tipki BTN3 na razvojni plošči Basys 2.



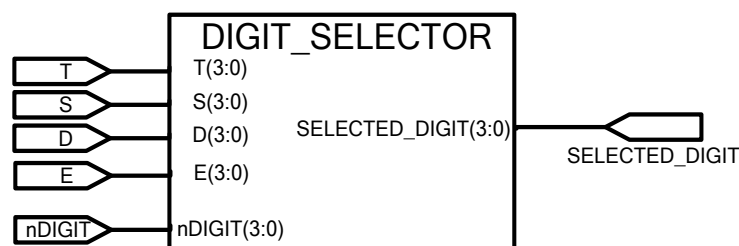
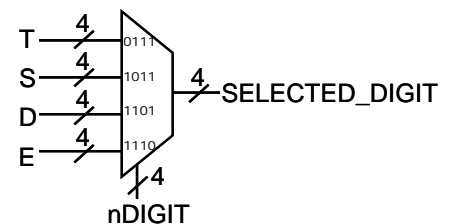
```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity led4digit is
    Port (CLK : in STD_LOGIC;
          nDIGIT : out STD_LOGIC_VECTOR (3 downto 0);
          LED : out STD_LOGIC_VECTOR (6 downto 0);
          BIN : in STD_LOGIC_VECTOR (3 downto 0)
        );
end led4digit;
architecture arch of led4digit is
begin
end arch;

```

- 2.4 Ustvarite nov projekt (Vaja2\_4), v katerem z izbirnim stavkom (**with select**) realizirajte vezje izbiralnika mest 4-mestnega LED prikazovalnika (DIGIT\_SELECTOR.vhd). Vezje izbiralnika izbira med štirimi 4-bitnimi vektorskimi vhodi (T, S, D, E), ki jih priredite 4-bitnemu vektorskemu izhodu (SELECTED\_DIGIT) glede na glede krmilni vhod (nDIGIT) po spodnji shemi. Če je vhod (nDIGIT = "0111"), potem na izhod pošljite vrednost tisočic (T). Ostale vrednosti so povzete v spodnji tabeli. Ta vaja nima UCF datoteke, zato jo lahko izdelate že doma in preizkusite njeno delovanje na simulatorju, tako da ustvarite nabor testnih vrednosti (DIGIT\_SELECTOR\_tb.vhd) v katerem preverite izide za vse vrednosti spodnje tabele.

vhod (nDIGIT)	izhod (SELECTED_DIGIT)	opomba
"0111"	T	-- <i>tisoci ce</i>
"1011"	S	-- <i>stoti ce</i>
"1101"	D	-- <i>deseti ce</i>
"1110"	E	-- <i>eni ce</i>

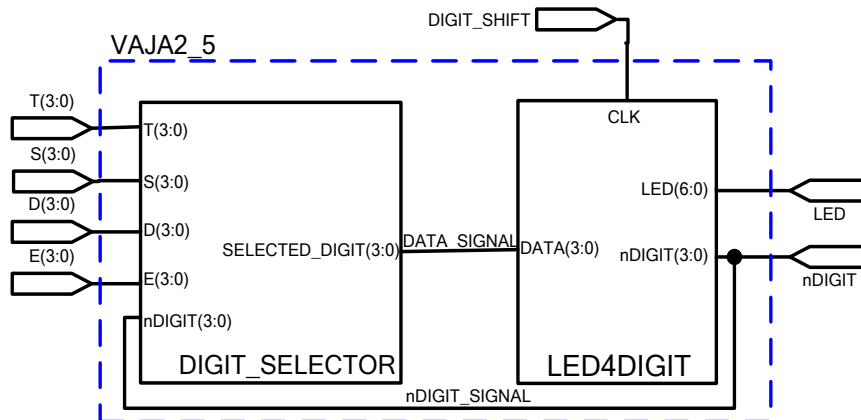


```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- T -> tisoci ce (thousands)
-- S -> stotice (hundreds)
-- D -> desetice (tens)
-- E -> eni ce (ones)
entity DIGIT_SELECTOR is
    Port (
        SELECTED_DIGIT : out STD_LOGIC_VECTOR (3 downto 0);
        T, S, D, E : in STD_LOGIC_VECTOR (3 downto 0);
        DIGIT : in STD_LOGIC_VECTOR (3 downto 0);
    );
end DIGIT_SELECTOR;
architecture arch of DIGIT_SELECTOR is
begin
end arch;

```

2.5 Ustvarite nov projekt (Vaja2\_5), v katerem povežite vezji izbiralnika mest (DIGIT\_SELECTOR.vhd) in krmilnika (led4digit.vhd) 4 mestnega LED prikazovalnika v novi datoteki (leddriver.vhd) po spodnji shemi. To datoteko bomo potrebovali vedno, ko bomo želeli izpisati neko štirimestno število (TSDE) na LED prikazovalnik. Ta vaja nima UCF datoteke, zato jo lahko izdelate že doma in preizkusite njeno delovanje na simulatorju, tako da ustvarite nabor testnih vrednosti (leddriver\_tb.vhd) v katerem preverite izide za vse vrednosti spodnje tabele.



```
entity leddriver is
    Port (LED : out STD_LOGIC_VECTOR(6 downto 0);
          nDIGIT : out STD_LOGIC_VECTOR(3 downto 0);
          T, S, D, E : in STD_LOGIC_VECTOR(3 downto 0);
          DIGIT_SHIFT : in STD_LOGIC);
end leddriver;
```

2.6 Ustvarite nov projekt (Vaja2\_6), v katerem na vhode (T, S, D, E) komponente (leddriver.vhd) povežete štiri dvobitne vrednosti s stikal SW0 do SW7. Velikost vhodov (T, S, D, E) je (3 downto 0), medtem ko imamo samo dve DIP stikali na eno mesto LED prikazovalnika, zato vhodom prilepite dve ničli na zgornji mesti (recimo T\_SIGNAL <= "00" & BIN(7) & BIN(6)) in na vhod komponente vodite celotni vmesni signal (recimo T\_SIGNAL). Spreminjajte dvobitne vhodne vrednosti na mestih tisočic, stotic, desetic in enic na stikalih SW0 do SW7 ter z vhodom (DIGIT\_SHIFT) na tipki BTN3 rotirajte prižiganje ustreznih LED prikazovalnikov. Če tipko BTN3 pritisnete hitro, se zdi kot da LED prikazovalnik naenkrat prikazuje vsa 4 mesta.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity vaj a2_6 is
    Port (LED : out STD_LOGIC_VECTOR(6 downto 0);
          nDIGIT : out STD_LOGIC_VECTOR(3 downto 0);
          DIP : in STD_LOGIC_VECTOR(7 downto 0);
          DIGIT_SHIFT : in STD_LOGIC);
end vaj a2_6;
architecture arch of vaj a2_6 is
begin
end arch;
```

