

RAZVOJ DIGITALNIH SISTEMOV

VAJA 4: PRETVORNIK IZ DVOJIŠKE KODE V BCD

4.1 Ustvarite nov projekt (Vaja4_1) in realizirajte ADD3 vezje v novi datoteki ADD3.vhd. Vezje naj izhod tvori tako, da vrednosti vhoda (A) prišteje 3, če je vhodna vrednost večja od 4, sicer naj bo izhod vezja (S) enak vhodu, kot prikazuje spodnja tabela. Vezje izdelajte na kombinacijski način z enim izbirnim stavkom (**when – else**), ki ga tvorite glede na podani opis delovanja.

Ta vaja sicer ima UCF datoteko, vendar jo lahko izdelate že doma in preizkusite njeno delovanje na simulatorju, tako da ustvarite nabor testnih vrednosti (ADD3_tb.tbw) v katerem preverite izide za vse vrednosti spodnje tabele.

A(3)	A(2)	A(1)	A(0)	S(3)	S(2)	S(1)	S(0)
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

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

entity ADD3 is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
          S : out  STD_LOGIC_VECTOR (3 downto 0));
end ADD3;

architecture arch of ADD3 is
begin
end arch;
```

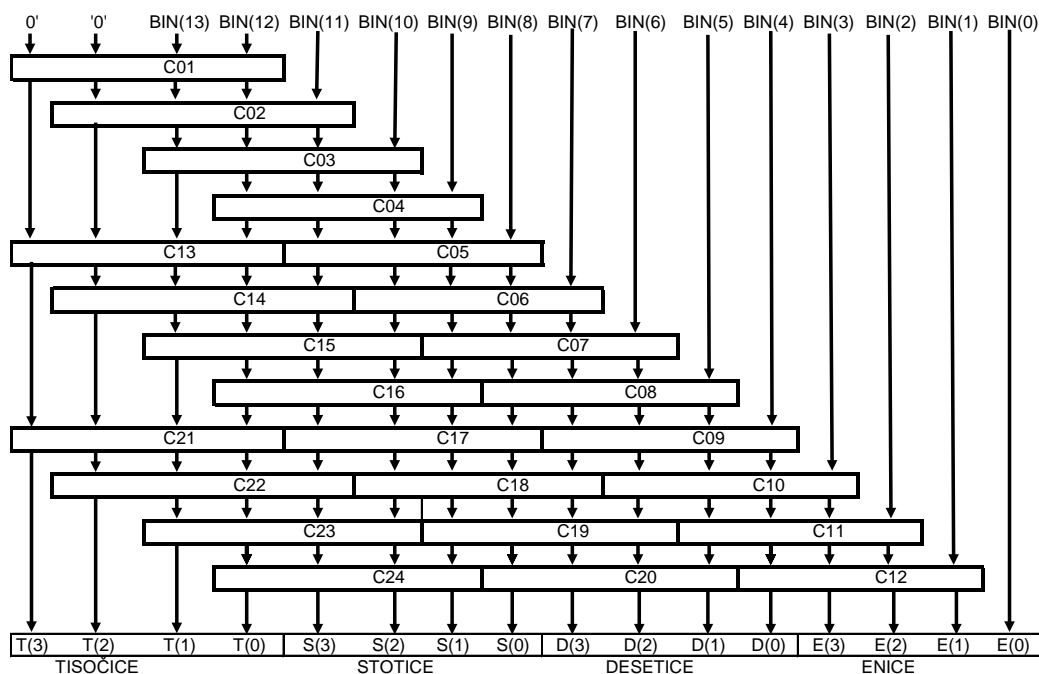
4.2 Ustvarite nov projekt (Vaja4_2) v katerem povežite komponente ADD3 (ADD3.vhd) 14 bitne BIN→BCD pretvorbe po algoritmu "double-dabble" z uporabo povezovalnega stavka (**port map**) v novi datoteki (bin2bcd.vhd). Imena komponent ADD3 naj bodo enaka kot na spodnji shemi (c01, c02, ... c24). Vezje ima 14-bitni vhodni vektorski signal (BIN) in tri 4-bitne izhodne vektorske signale za tisočice, stotice, desetice in enice (T, S, D, E). Vsaka komponenta ADD3 ima že definiran lasten vhodni (cXi n) in izhodni signal (cXout) za povezovanje, kjer je X številka modula 01 do 24. Primer povezovanja vhoda bloka C03:

`c03in <= c02out(2 downto 0) & BIN(10);`

Izhodne signale blokov, ki se povezujejo na izhodne signale (T, S, D, E) v shemi povežite neposredno. Primer povezovanja vektorskega izhodnega signala tisočic (T):

`T <= c21out(3) & c22out(3) & c23out(3) & c24out(3);`

Na vhod MSB mesta in mesto nižje postavite logično '0'. Ta vaja nima UCF datoteke, zato jo lahko izdelate doma in preizkusite njeno delovanje na simulatorju, tako da ustvarite nabor testnih vrednosti (bin2bcd_tb.tbw) v katerem preverite pravilnost delovanja za primer: $270F_{16} \rightarrow 9999_{10}$.



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

entity BIN2BCD is
    Port ( BIN : in  STD_LOGIC_VECTOR (13 downto 0);
          T, S, D, E : out STD_LOGIC_VECTOR (3 downto 0));
end BIN2BCD;
```

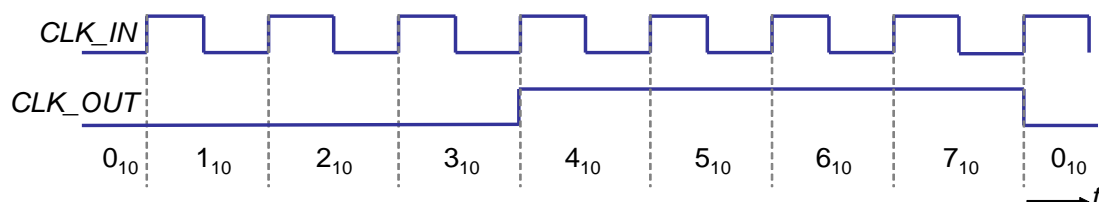
```
architecture arch of BIN2BCD is
    signal c01in, c02in, c03in, c04in, c05in, c06in, c07in, c08in, c09in, c10in
    : STD_LOGIC_VECTOR (3 downto 0);
    signal c11in, c12in, c13in, c14in, c15in, c16in, c17in, c18in, c19in, c20in
    : STD_LOGIC_VECTOR (3 downto 0);
    signal c21in, c22in, c23in, c24in : STD_LOGIC_VECTOR (3 downto 0);
    signal c01out, c02out, c03out, c04out, c05out, c06out, c07out, c08out,
    c09out, c10out: STD_LOGIC_VECTOR (3 downto 0);
    signal c11out, c12out, c13out, c14out, c15out, c16out, c17out, c18out,
    c19out, c20out: STD_LOGIC_VECTOR (3 downto 0);
    signal c21out, c22out, c23out, c24out : STD_LOGIC_VECTOR (3 downto 0);
begin
end arch;
```

4.3 Ustvarite nov projekt (Vaja4_3) in realizirajte delilnik frekvence signala ure (CLOCK_DIVIDER.vhd), katerega izhodna frekvenca bo 0,4 Hz, tako da bomo na LED diodi opazovali utripanje. Vhodno frekvenco ure 50 MHz moramo tako zmanjšati za faktor 125000000. Vezje ima vhod na katerega bomo vodili visoke vhodne frekvence (CLK_IN) in izhod upočasnjenega signala ure (CLK_OUT). V UCF datoteki je izhod upočasnjenega signala ure (CLK_OUT) vezan na diodo LED1. Tipka za ponastavitev (nRST) se nahaja na BTN0. Spodaj je prikazana je VHDL realizacija delilnika frekvence z uporabo števca. Navedeni primer predelajte za potrebe delilnika frekvence iz naloge.

```
entity CLOCK_DIVIDER is
    generic( Clock_divisor : natural := 20000000
    );
    Port (nRST, CLK_IN : in STD_LOGIC;
          CLK_OUT : out STD_LOGIC);
end CLOCK_DIVIDER;

architecture arch of CLOCK_DIVIDER is
    signal COUNTER: STD_LOGIC_VECTOR (15 downto 0) := X"00000000";
begin
    process (nRST, CLK_IN, COUNTER)
    begin
        if (nRST = '0') then
            COUNTER <= (others => '0');
        elsif ( rising_edge(CLK_IN) ) then
            if( COUNTER = Clock_divisor ) then
                COUNTER <= (others => '0'); -- set COUNTER to 0
            else
                COUNTER <= COUNTER + 1; -- add '1'
            end if;
        end if;
    end process;
    CLK_OUT<= '0' when (COUNTER < (Clock_divisor / 2)) else '1'; -- duty
cycle
end arch;
```

Zgornji procesni stavek realizira 16 bitni števec navzgor, ki po vklopu vezja šteje od 0_{10} do vključno 7_{10} . Funkcija (`rising_edge(CLK_IN)`) pomeni, da se prištevanje 1 (štetje) izvaja na prednji rob signala ure (CLK_IN). Čim števec prešteje do vrednosti modula števca (`Clock_divisor`), se postavi nazaj na 0_{10} . Kombinacijsko (izven procesa) postane izhod upočasnjenega signala ure (CLK_OUT) '0', če je vsebina števca manjša od (`Clock_divisor/2`), sicer je '1'. S tem stavkom določamo razmerje signal/pavza oz. razmerje kolikšen del periode bo izhodni signal '1' in kolikšen del bo '0' (ang. duty cycle). Delovanje vezja ilustrira spodnja slika v katerem je razmerje signal/pavza = 50%.



Slika 1: Štetje po modulu 8 z razmerjem signal/pavza 50%.

Če želite dodatno upočasniti izhodni signal (CLK_OUT), mora števec šteti do višje vrednosti. Pomembno je, da pri določanju modula štetja (najvišja vrednost štetja + 1) v deklaraciji signala števca (COUNTER) rezervirate dovolj mest za zapis višjega modula štetja.

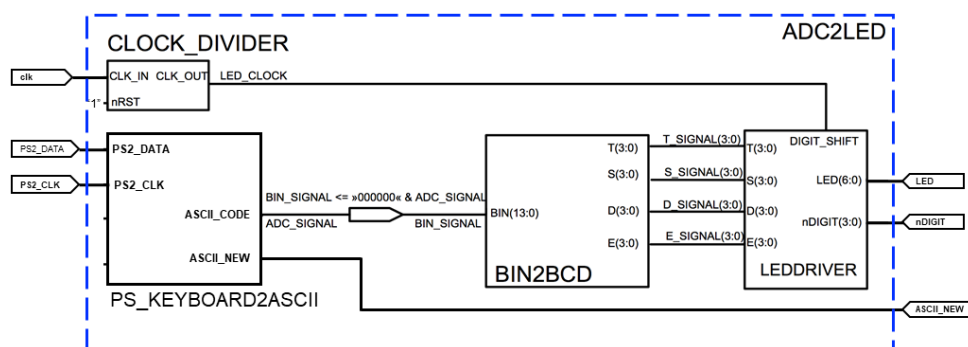
4.4 Ustvarite nov projekt (vaja4_4) in komponente iz prejšnjih nalog povežite s povezovalnimi stavki na gonilnik za LED prikazovalnik po spodnji shemi v novi datoteki (ADC2LED.vhd). Gonilnik za LED prikazovalnik (1eddri ver. vhd) smo izdelali v vaji 2.5, zato kopirajte vse VHD datoteke (brez UCF) iz mape 2.5 v mapo vaje 4.4 enako pa storite z datotekami iz vaje 3.3 za tipkovnico.

Izhodni signal iz tipkovnice vsebuje 7 bitno ASCII kodo, vhod za pretvornik kode BIN→BCD je 14-biten, zato vsa višja mesta pretvornika kode postavite na '0' z uporabo operatorja lepljenja ($BIN_SIGNAL \leq "0000000" \& ADC_SIGNAL$). Rezultat pretvorbe vodimo na pretvornik dvojiške kode v BCD zapis. Izhode pretvornika (T, S, D, E), vodimo na krmilnik prikazovalnika (1eddri ver. vhd).

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

entity ADC2LED is
    Port (
        nDIGIT : out STD_LOGIC_VECTOR (3 downto 0);
        LED: out STD_LOGIC_VECTOR (6 downto 0);
        clk : in STD_LOGIC;
        PS2_CLK : in STD_LOGIC;
        PS2_DATA : in STD_LOGIC;
        ASCII_NEW : out STD_LOGIC);
end ADC2LED;

architecture arch of ADC2LED is
begin
end arch;
```



Slika 2: Povezovanje BIN→BCD pretvornika s PS/2 tipkovnico.