9. Vaja: komunikacija med AVR in CPLD

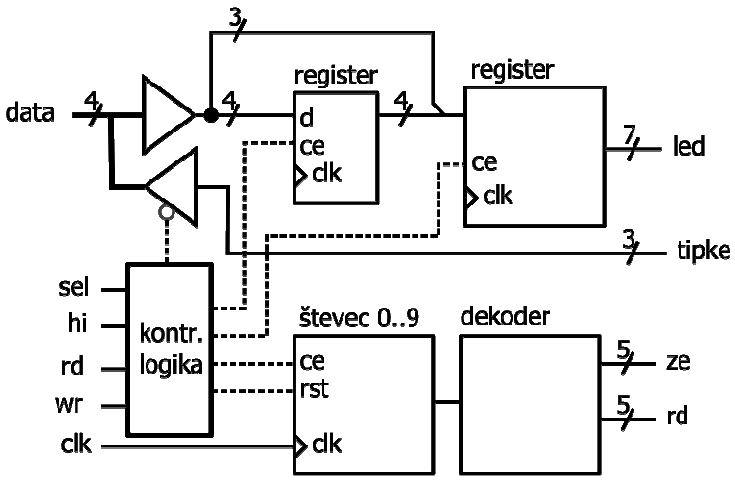
9.1 Naloga (VHDL)

Naredi vmesnik za krmiljenje LED matrike s procesorjem AVR, ki je na razvojnem sistemu Arduino Nano. Vodilo med procesorjem in programirljivim vezjem sestavljajo signali:

- data(3:0), 4 bitno dvosmerno vodilo za podatke
- wr, ob impulzu na '0' procesor vpisuje podatek na vodilo
- rd, ob impulzu na '0' procesor bere podatek
- hi, kadar je '0' so na vodilu zgornji 4 biti, sicer pa spodnji biti 8-bitne besede
- sel, kadar je '0' vpisujemo prvi podatek (reset števca), sicer pa naslednje podatke

Vmesnik naj vsebuje števec in dekodirnik, ki zaporedno izbira stolpce matrike po tabeli:

števec	**ze**	**rd**
0	00001	00000
1	00010	00000
2	00100	00000
3	01000	00000
4	10000	00000
5	00000	00001
6	00000	00010
7	00000	00100
8	00000	01000
9	00000	10000



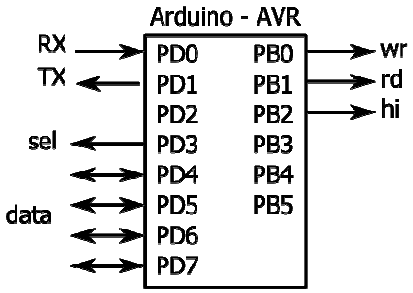
Podatki se prenesejo na izhod LED v dveh fazah:

- pri hi='0' in wr='0' se vpišejo 4 biti iz vodila data v prvi register
- pri hi='1' in wr='0' se vpišejo 3 biti iz vodila data na spodnje bite drugega registra LED(2 **downto** 0), hkrati pa se prepiše podatek iz prvega registra na LED(6 **downo** 3)

9.2 Naloga (programiranje)

Namesti knjižnico MsTimer2, ki omogoča izvajanje funkcije ob izteku milisekundnega časovnika. Poglej priložen primer kode za utripanje LED v taktu 500 ms.

Vsebina LED matrike naj bo v globalni spremenljivki, ki je deklarirana kot polje velikosti 10 bajtov (podatkovni tip **unsigned char**). Posamezen bajt naj predstavlja en stolpec, najprej so stolpci zelenih, nato pa še stolpci rdečih LED.



Napiši funkcijo, ki ob vsakem klicu izvede prenos podatkov na en stolpec LED matrike. Številka stolpca naj bo v lokalni statični spremenljivki in naj se ob vsakem klicu poveča za 1. Matrika ima 10 stolpcev. Funkcija naj se z uporabo časovnika izvede vsake 2ms (v funkciji nastavi vrednost 1, ker je frekvenca ure pol nižja od pričakovane). Prenos podatka poteka po korakih:

1. Na vratih PD nastavi zgornjih 5 bitov kot izhode (DDRD=…)
2. Preberi iz polja en stolpec in pošlji zgornje 4 bite na PORTD. Če vpisujemo prvi stolpec, naj bo PD3 sel=0, sicer pa mora biti sel=1.
3. Naredi na PORTB dogodek za vpis zgornjih bitov podatka
    a. nastavi hi=0, rd=1, wr=0
    b. nastavi hi=0, rd=1, wr=1
4. Pomakni podatkovne bite za 4 mesta v levo in jih pošlji na PORTD. Če vpisujemo prvi stolpec, naj bo PD3 sel=0, sicer pa mora biti sel=1.
5. Naredi na PORTB dogodek za vpis spodnjih bitov podatka
    a. nastavi hi=1, rd=1, wr=0
    b. nastavi hi=1, rd=1, wr=1
6. Za branje vrednosti tipk najprej nastavi zgornje 4 bite na PD kot vhod (DDRD=…)
7. Pošlji na PORTB ukaz za branje hi=1, rd=0, wr=1 in naredi kratko pavzo pred branjem podatka (npr. vpiši v zgornje 4 bite PORTD vrednost 0)
8. Preberi podatek iz PD (tipka = PIND…) in ga shrani v globalno spremenljivko .
9. Zaključi ukaz za branje hi=1, rd=1, wr=1

Pri pisanju programa uporabljaj direkten dostop do vrat (globalne spremenljivke DDRD, PORTD, PIND…), ki je hitrejši kot klic namenskih funkcij. V funkciji setup() nastavi začetno stanje na PORTD in PORTB – najbolj pomembno je da sta v začetnem stanju wr=1 in rd=1, ker vrednost 1 predstavlja neaktivni dogodek.