Vaja 9

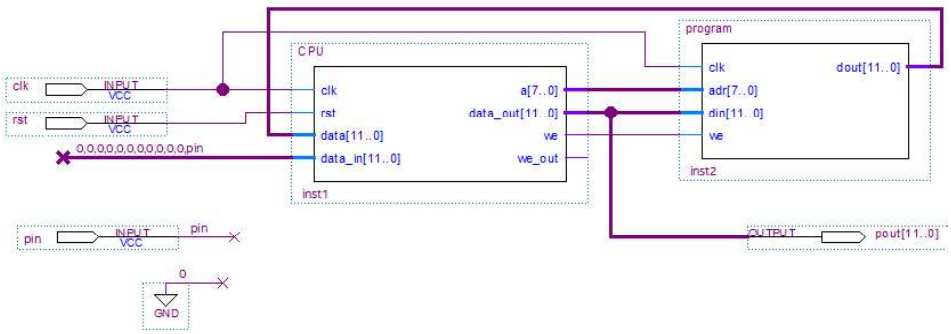
Mikroprocesor CPE4

Pri vaji boste uporabljali učni mikroprocesor, ki je narejen z logiko v vezju FPGA. Pomemben del načrtovanja takšnega vezja je načrtovanje kode, ki naj teče na procesorju. V projekt boste kodo v zbirnem jeziku vnesli s pomočjo spletnega orodja: https://lniv.fe.uni-lj.si/cpu.html, s katerim je mogoče izvajanje programov tudi simulirati.

9.1 Gradnik: procesorski blok

Procesorski blok temelji na majhni 12-bitni centralno procesni enoti CPE4, ki je narejena v strojno-opisnem jeziku VHDL. Procesor pozna 2^4 različnih strojnih ukazov za obdelavo 12-bitnih podatkov. Procesorski blok ima 12-bitno vhodno in izhodno vodilo za povezavo v digitalni sistem.

Poleg CPE4 je v bloku še pomnilnik RAM v katerem so shranjeni strojni ukazi in spremenljivke. Tudi pomnilnik je opisan v jeziku VHDL in ob prevajanju vezja že vsebuje program za mikroprocesor.



Slika 9.1: *Procesorski blok s centralno procesno enoto in pomnilnikom.*

Slika 9.1 prikazuje gradnik ProcBlok, ki je sestavljen iz centralne procesne enote in pomnilnika. Gradnik ProcBlock ima naslednje priključke:

- **clk** - vhodna ura (procesor je sekvenčno vezje!),
- **rst** - signal reset, ob 1 se procesor ponastavi na začetek programa (naslov 0),
- **pin** - vhodno vodilo, na katerega je povezan le en bit in
- **pout [11..0]** - 12-bitno izhodno vodilo.

Procesor in pomnilnik sta opisana v datotekah:

- *proc.vhd* - opis procesorskega jedra.
- *procpak.vhd* - definicije zbirniških ukazov, ki jih pozna procesor.
- *program.vhd* - opis pomnilnika in njegova vsebina. To datoteko urejate, ko želite v procesor vpisati program.

9.2 Program v zbirnem jeziku

Na spletni strani: `https://lniv.fe.uni-lj.si/cpu.html` je urejevalnik zbirne kode in simulator 12-bitnega procesorja. Procesor je zelo enostaven in pozna le 16 ukazov s katerimi prenašamo podatke, računamo in izvajamo skoke. Podatki prihajajo iz pomnilnika (spremenljivke) ali pa vhodnega vodila. Rezultat ukazov se shrani v register z imenom **akumulator**.

Ukaze zapisujemo v zbirniku vsakega v svojo vrstico, ki se začne z oznako ali pa s presledkom (oz. tabulatorjem). Oznake potrebujemo za skočne ukaze, npr. ukaz **jmp start** bo izvedel skok na ukaz, pred katerim stoji oznaka **start:**

Poglejmo primer majhnega programa, ki v neskončni zanki bere vrednost iz vhoda, jih prišteje spremenljivki **n** in shrani na izhod:

```
start:  inp vh
        add n
        sta n
        outp izh
        jmp start
vh      di 0
izh     do 0
n       db 0
```

V prvi vrstici beremo podatek iz vhodnega vodila, v naslednji pa mu prištejemo vrednost spremenljivke **n**. Rezultat bo v akumulatorju, zato potrebujemo še ukaz **sta**, s katerim shranimo vrednost spremenljivke, ki se nahaja v pomnilniku.

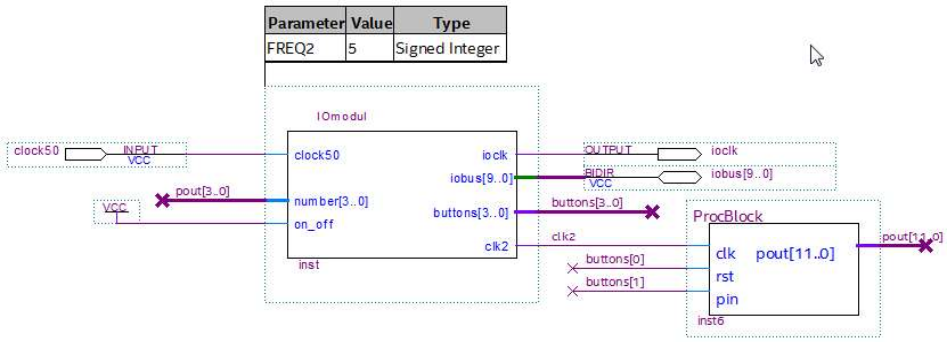
S stavkom **outp** prenesemo vrednost na izhodno vodilo, da jo bomo lahko opazovali na prikazovalniku. Program zaključimo s skokom na začetek, ki povzroči ponavljanje opisanega algoritma. Procesor stalno izvaja ukaze, saj nima ukaza za zaustavitev ali spanje, zato mora biti program vedno napisan v zanki.

Na koncu kode so deklaracije vhodnih spremenljivk **di**, izhodnih spremenljivk **do** in notranjih spremenljivk programa **db**. Številka za oznako **db** določa začetno vrednost spremenljivke; v našem primeru ima spremenljivka **n** začetno vrednost 0.

Delovanje programa preizkušamo v simulatorju, kjer napišemo zbirno kodo, jo prevedemo (**Start**) in izvajamo po korakih (**Korak**). Med izvajanjem nastavljamo vrednosti vhoda ter opazujemo akumulator in vsebino pomnilnika.

9.3 Shema procesorskega sistema

Pri vaji boste dobili že narejeno shemo sistema, kjer je procesorski blok povezan na IOmodul in LED. Prva tipka je povezana na signal reset, druga pa na vhodno vodilo. Procesor je povezan na počasno uro **clk2** s frekvenco 5 Hz, da bomo na vezju lažje opazovali spreminjanje izhodov. Ura in tipke so veznane na LED, izhodno vodilo procesorja pa na matrični prikazovalnik (opazujemo le najnižje štiri bite).



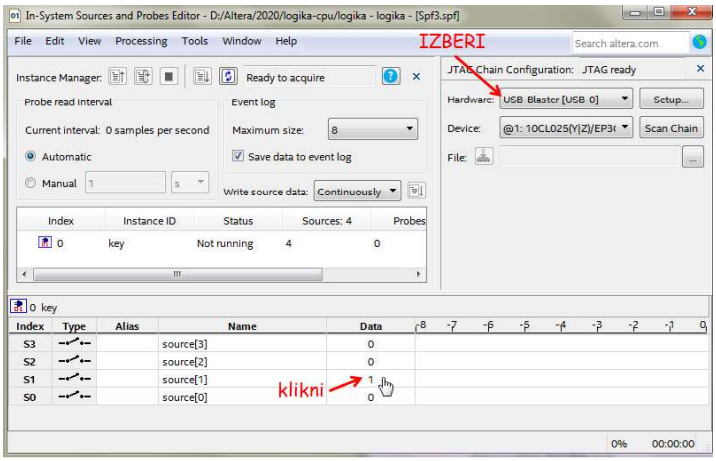
Slika 9.2: *Sistem z vmesnikom in procesorskim blokom.*

9.4 Kaj morate narediti vi?

Najprej preizkusite delovanje procesorja na simulatorju. Prepišite program v urejevalnik in na spletnem simulatorju opazujte izvajanje programa po korakih. Nastavite vrednost vhodnega vodila na 1, tako da v okencu Vhodi vnesete vrednost 1 desno od vh. Opazujte, kako se spreminja vrednost v akumulatorju.

Odprite arhiv programa Quartus v katerem je digitalni sistem s procesorjem. Preglejte glavno shemo logika.bdf in shemo procesorskega bloka, ki jo lahko odprete z desnim klikom in Open Design File.

Prevedite sistem in preizkusite delovanje na razvojni plošči. Pritisnite tipko S2 oz. nastavite virtualno tipko source[1] (slika 9.3) in opazujte kako se spreminja vrednost na matričnem LED.



Slika 9.3: *Nastavitev virtualnih tipk: izberi Tools, In-System Sources, nastavi Hardware: USB Blaster in klikni Data pri ustrezni tipki.*

Ura procesorja je nastavljena na 5 Hz. Če želite, jo spremenite z nastavitvijo parametra FREQ2 v shemi vezja. Odprite datoteko program.vhd in preglejte vsebino pomnilnika:

```
18  signal m : memory := (
19  -- sem pisete vas program
20   0=> lda & x"08",
21   1=> sta & x"07",
22   2=> inp & x"00",
23   3=> add & x"07",
24   4=> sta & x"07",
25   5=> outp & x"00",
26   6=> jmp & x"02",
27   7=> x"000",
28   8=> x"000",
29   -- tu se konca vas program
30   others => x"000"
31   );
```

Na matričnem prikazovalniku opazujemo vrednost izhodnega vodila, ki se spreminja med izvajanjem programa. Na vodilo je povezana kar vsebina akumulatorja. Po branju iz vhoda (**lda**) je v akumulatorju 0 ali 1, po prištevanju spremenljivke **n** pa ustrezna vsota. Če želimo, da bo na izhodu prikazana le vsota, moramo v vezje dodati register. Z registrom bomo izhodno vodilo spremenili v izhodna vrata (angl. port), ki spreminjajo vrednost le ob izvedbi ukaza **outp**.

Na shemo procesorskega bloka ProcBlock.bdf dodajte 12-bitni register reg12. Povežite podatkovni vhod na izhodno vodilo procesorja (data_out) in izhod na zunanje priključke (pout). Register naj ima uro povezano na priključek clk in signal load na we_out (ta signal se postavi, ko procesor piše na zunanje vodilo). Prevedite sistem in preizkusite delovanje na razvojni plošči.

Razmisli

Opazuj, kako deluje signal reset (prva tipka). Koda iz poglavja 8.2 in vsebina programskega pomnilnika se nekoliko razlikujeta, saj sta v pomnilniku na začetku še dva dodatna ukaza. Razloži zakaj ju potrebujemo.