

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Digitalna logika

Laboratorijske vježbe korištenjem sklopovskih pomagala

Upute za 5. laboratorijsku vježbu

Marko Zec

Prosinac 2017.

1 Uvod

Cilj vježbe je upoznavanje s temeljnim memorijskim elementima i njihovim opisivanjem jezikom VHDL kroz primjere povezivanja sinkronih, bridom okidanih registara i kombinacijske logike u jednostavne sekvencijske sklopove – brojila.

Kao ilustracija i podloga za kasniju samostalnu izradu zadatka pripremljen je predložak `broj.vhd` koji je dostupan i na web sjedištu laboratorijskih vježbi:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

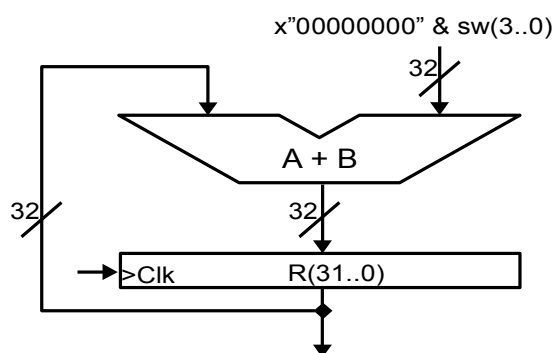
entity broj is
  port (
    clk_25m: in std_logic;
    btn_down: in std_logic;
    sw: in std_logic_vector(3 downto 0);
    led: out std_logic_vector(7 downto 0)
  );
end broj;

architecture x of broj is
  signal R, delta: std_logic_vector(31 downto 0);
begin
  process(clk_25m)
  begin
    if rising_edge(clk_25m) then
      R <= R + delta;
    end if;
  end process;

  delta <= x"00000000" & sw;

  led <= R(31 downto 24) when btn_down = '0' else R(23 downto 16);
end x;
```

Sklop broj opisan VHDL-in je 32-bitno binarno brojilo čija je temeljna struktura prikazana slikom 1:



Slika 1: struktura 32-bitnog brojila

Uvjet za upis nove vrijednosti u registar `R` zadan je unutar bloka `process`, što je standardni obrazac za opis bistabila odnosno registara VHDL-om. Alternativno, većina modernih alata za sintezu prihvatit će i kraći oblik opisa registra kroz konkurentni izraz (bez bloka `process`):

```
R <= R + delta when rising_edge(clk_25m);
```

32-bitna vrijednost `delta` za koju se u svakom ciklusu takta `clk_25m` povećava broj pohranjen u registru `R` dobija se proširenjem 4-bitnog ulaza `sw` na 32 bita, dopunjavanjem nulama na pozicijama bitova više težine.

Stanje bitova više težine registra `R` može se pratiti na LED indikatorima. Zavisno od stanja ulaza povezanog s tipkom `btn_down`, multipleksor će na LED indikatore propuštati bitove 31..24, odnosno bitove 23..16.

2 Zadaci

2.1 Sinteza i ispitivanje sklopa iz predloška "broj"

Stvorite novi projekt u razvojnom okruženju Lattice Diamond i u njega uključite modul `broj.vhd` dohvaćen s web sjedišta laboratorijskih vježbi. Sintetizirajte konfiguracijsku datoteku i programirajte FPGA sklop.

Ispitajte rad sklopa opažanjem odziva LED indikatora uz različite postavke prekidača razvojne pločice.

Modificirajte opis sklopa zamjenom bloka `process` konkurentnim izrazom s prethodne stranice. Sintetizirajte sklop i ispitajte njegov odziv na različitu pobudu (prekidači, tipka).

2.2 Proširenje sklopa: *write enable*

Proširite sklop dodavanjem upravljačkog ulaza za dozvolu upisa u registar (*write enable*), kojeg povežite s proizvoljno odabranom tipkom razvojne pločice. Sintetizirajte sklop i ispitajte njegovu funkcionalnost.

Prošireni sklop pohranite u sustav Ferko pod imenom `broj_ce.vhd`.

2.3 8-bitno brojilo

Konstruirajte 8-bitno binarno brojilo kojem se u svakom ciklusu takta vrijednost povećava za jedan. Kao izvor signala takta koristite neku od tipki, npr. `btn_down`. Stanje registra proslijedite na LED indikatore. Sklop opišite samo jednim VHDL modulom odnosno datotekom, bez instanciranja dodatnih komponenti. Registar možete opisati ili blokom `process` ili konkurentnim izrazom, kako Vam se više sviđa.

Sintetizirajte i ispitajte sklop. Broji li brojilo pouzdano baš sve otkucaje tipke?

2.4 Proširenje 8-bitnog brojila: filtriranje signala takta

Proširite sklop 8-bitnog brojila instanciranjem modula `upravljac` iz 4. laboratorijske vježbe, na način da se upis u registar brojila više ne okida direktno signalom iz tipke razvojne pločice, nego filtriranim signalom koji dolazi iz modula `upravljac`.

Predložak za instanciranje modula potražite u modulu `datapath` iz 4. laboratorijske vježbe. Svom sklopu morat ćete dodati vanjski ulaz `clk_25m`, kojeg treba sprovesti do istoimenog ulaza modula `upravljac`. Potrebno je deklarirati i dodatni interni signal `clk`

kojeg ćete sprovesti s izlaza modula `upravljac` i koristiti za okidanje upisa u registar brojila. Ulaze i izlaze modula `upravljac` koje ne koristite možete povezati na nulu, odnosno deklarirati ih kao nepovezane (ključna riječ `open`):

```
port map (  
    clk_25m => clk_25m, Clk_key => btn_down, Clk => clk,  
    AddrA_key => '0', AddrB_key => '0', ALUOp_key => '0',  
    AddrA => open, AddrB => open, ALUOp => open  
);
```

Sintetizirajte i ispitajte sklop. Broji li sad brojilo pouzdano otkucaje tipke?

Prošireni sklop pohranite u sustav Ferko pod imenom `broj8.vhd`.

2.5 Proširenje 8-bitnog brojila: asinkroni ulaz za *reset*

Proširite sklop 8-bitnog brojila ulazom za asinkroni *reset*, za kojeg signal dovedite s proizvoljne tipke razvojne pločice. Sintetizirajte sklop i ispitajte njegovu funkcionalnost.

Prošireni sklop pohranite u sustav Ferko pod imenom `broj8_ar.vhd`.

2.6 Proširenje 8-bitnog brojila: sinkroni ulaz za *reset*

Promijenite opis sklopa na način da ulaz za *reset* djeluje sinkrono. Sintetizirajte sklop i ispitajte njegovu funkcionalnost.

Prošireni sklop pohranite u sustav Ferko pod imenom `broj8_sr.vhd`.

2.7 Proširenje 8-bitnog brojila: brojanje u modulu N

Proširite sklop 8-bitnog brojila iz prethodnog koraka na način da broji u modulu N. Drugim riječima, brojlo nakon dosezanja vrijednosti $N - 1$ u slijedećem ciklusu takta mora poprimiti vrijednost nula.

N je broj određen sa zadnje dvije znamenke Vašeg JMBAG identifikatora, u dekadskom zapisu. Ako je predzadnja znamenka Vašeg JMBAG identifikatora jednaka nuli, broj N formirajte kao $10 +$ zadnja znamenka JMBAG identifikatora.

Sintetizirajte sklop i ispitajte njegovu funkcionalnost. Prošireni sklop pohranite u sustav Ferko pod imenom `broj8_modn.vhd`.