Univerzitet u Nišu
Elektronski fakultet
Katedra za Računarstvo

Arhitektura i organizacija računara

VHDL opis kompleksinijih primera sa časova računskih vežbi

U ovom dokumentu dati su opisi dela primera sa časova računskih vežbi.

Materijal je namenjen za lakše praćenje računskih vežbi i potrebno ga je poneti na nastavu u obliku da po njemu može da se beleži.

Materijal nije namenjen za samostalno izučavanje predmetne materije. Kolekcija primera je delimična i ne sadrži sve primere sa časova, teoretsku podlogu, objašnjenja, komentare, crteže, alternative i diskusiju rešenja, a može da sadrži namerne (i/ili nenamerne) greške. Svi ovi dodatni elementi će biti dati na časovima računskih vežbi i samo uz njih se može dobiti potpun materijal pogodan za učenje.

- Deo 3 -

Primer 17. Sistem sa memorijom i aritmetikom

Koristi kola iz primera 9, 10, 14, 15,

```
--Upravljacka logika
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY Kontrola IS
  PORT ( clk, WE, reset : IN STD_LOGIC;
         addr : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
         rstAddr: OUT std_logic;
         upis1  : OUT STD_LOGIC;
         upis2  : OUT STD_LOGIC);
END Kontrola;

ARCHITECTURE stateMachine OF Kontrola IS
```

```vhdl
  TYPE states IS (sWrite, sRead1, sRead2, sStart);
  SIGNAL current_state, next_state: states;
  SIGNAL upis1_async, upis2_async: STD_LOGIC;
BEGIN
  stateChange: PROCESS (reset, clk)
  BEGIN
        IF (reset='1') THEN
              current_state <= sStart;
        ELSIF (clk'EVENT AND clk='1') THEN
              upis1<=upis1_async;
              upis2<=upis2_async;
              current_state <= next_state;
        END IF;
  END PROCESS;
  OutputAndStateCalculation: PROCESS (WE, addr, current_state)
  BEGIN
        CASE current_state IS
              WHEN sStart =>
                    rstAddr<='1';
                    upis1_async<='0';
                    upis2_async<='0';
                    IF (WE='1') THEN
                          next_state <= sWrite;
                    ELSIF (WE='0') THEN
                          next_state <= sRead1;
                    END IF;
              WHEN sWrite =>
                    rstAddr<='0';
                    IF (WE='1') THEN
                          next_state <= sWrite;
                    ELSIF (WE='0') THEN
                          next_state <= sStart;
                    END IF;
              WHEN sRead1 =>
                    rstAddr<='0';
                    upis1_async <= '1';
                    upis2_async <='0';
                    IF (WE='0') THEN
                          next_state <= sRead2;
                    ELSIF (WE='1') THEN
                          next_state <= sStart;
                    END IF;
              WHEN sRead2 =>
                    upis1_async <= '0';
                    upis2_async <='1';
                    IF (WE='0') THEN
                          next_state <= sRead1;
                    ELSIF (WE='1') THEN
                          next_state <= sStart;
                    END IF;
        END CASE;
  END PROCESS;
END ARCHITECTURE stateMachine;

-- top level
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY sistem IS
  PORT( clk, WE, reset : IN std_logic;
        din : IN std_logic_vector(7 DOWNTO 0);
        dout : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
  );
END ENTITY sistem;
ARCHITECTURE structural OF sistem IS
  SIGNAL addr, data_int, operand1, operand2 : STD_LOGIC_VECTOR (7 DOWNTO
0);
  SIGNAL addrInt: integer RANGE 0 TO 255;
  SIGNAL rstAddr, upis1, upis2 : STD_LOGIC;

BEGIN
  cu: ENTITY work.Kontrola(stateMachine)
        PORT MAP(
              clk=>clk, WE=>WE, reset=>reset,
              addr => addr,
              rstAddr=>rstAddr,
              upis1=> upis1, upis2 => upis2
        );

  cnt: ENTITY work.counter8(counter8_arch)
        PORT MAP(
              clk=>clk,reset=>rstAddr,
              ce=>'1', load=>'0', dir=>'1',
    din=> 0,
    count=>addrInt);

  addr<=std_logic_vector(to_unsigned(addrInt, 8));

  mem: ENTITY work.Memorija(Behavioral)
        PORT MAP( WE =>WE, clk => clk, addr => addr,
              data =>din, Q =>data_int
        );

  buff1: ENTITY work.register_tristate(cell_level)
        GENERIC MAP(width=>8)
        PORT MAP(clock=>upis1,
              out_enable=>'1',
              data_in=>data_int,
              data_out=>operand1
        );

  buff2: ENTITY work.register_tristate(cell_level)
        GENERIC MAP(width=>8)
        PORT MAP(clock=>upis2,
              out_enable=>'1',
              data_in=>data_int,
              data_out=>operand2
        );

  sabirac: ENTITY work.carry_ripple_adder(w_generate)
        GENERIC MAP(n=>8)
        PORT MAP( a=>operand1, b=>operand2, cin=>'0',
              s=> dout(7 DOWNTO 0),
                 cout=> dout(8)
                   );
END ARCHITECTURE structural;
```