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 1 -

Primer 1. Jednobitni potpuni sabirač

```
ENTITY full_adder IS
   PORT (a, b, c_in: IN BIT; s, c_out: OUT BIT);
END ENTITY full_adder;
---
ARCHITECTURE truth_table OF full_adder IS
BEGIN
   WITH BIT_VECTOR'(a, b, c_in) SELECT
         (c_out, s) <=     BIT_VECTOR'("00") WHEN "000",
                           BIT_VECTOR'("01") WHEN "001",
                           BIT_VECTOR'("01") WHEN "010",
                           BIT_VECTOR'("10") WHEN "011",
                           BIT_VECTOR'("01") WHEN "100",
                           BIT_VECTOR'("10") WHEN "101",
                           BIT_VECTOR'("10") WHEN "110",
                           BIT_VECTOR'("11") WHEN "111";
END ARCHITECTURE truth_table;
```

Primer 2. Trobitni sabirač, strukturalni opis

```vhdl
ENTITY adder3b IS
   PORT (op1, op2: IN bit_vector(2 DOWNTO 0);
               cin: IN bit;
               sum: OUT bit_vector(2 DOWNTO 0);
               cout: OUT bit);
END ENTITY adder3b;

ARCHITECTURE struct OF adder3b IS
   SIGNAL c01, c12: bit;
BEGIN
   bit0: ENTITY work.full_adder(truth_table)
         PORT MAP(a=>op1(0), b=>op2(0), c_in=>cin,
                  s=>sum(0), c_out=>c01);
   bit1: ENTITY work.full_adder(truth_table)
         PORT MAP(op1(1), op2(1), c01, sum(1),c12);
   bit2: ENTITY work.full_adder(truth_table)
         PORT MAP(op1(2), op2(2), c12, sum(2),cout);
END ARCHITECTURE struct;
```

Primer 3. Testbench za trobitni sabirač

```vhdl
ENTITY adder3b_tb IS
END ENTITY adder3b_tb;
ARCHITECTURE tb OF adder3b_tb IS
   SIGNAL sigA,sigB,sigC : bit_vector(2 DOWNTO 0);
   SIGNAL c_in, c_out : bit;
BEGIN
   uut: ENTITY adder3b(struct)
         PORT MAP(
               op1=>sigA,
               op2=>sigB,
               cin=>c_in,
               sum=>sigC,
               cout=>c_out
         );
   stimuli: PROCESS
   BEGIN
         sigA<="001";
         sigB<="010";
         c_in<='0';
         WAIT FOR 1 ns;
         sigA<="111";
         sigB<="010";
         c_in<='0';
         WAIT FOR 1 ns;
         sigA<="111";
         sigB<="010";
         c_in<='1';
         WAIT FOR 1 ns;
         --...
   END PROCESS stimuli;
END ARCHITECTURE tb;
```

Primer 4. D flip-flop sa asinhronim i sa sinhronim resetom

```vhdl
ENTITY edge_triggered_Dff IS
   PORT (      d: IN bit; clk: IN bit; clr: IN bit;
         q: OUT bit;);
```

```vhdl
END ENTITY edge_triggered_Dff;
-------
ARCHITECTURE asyncCLR OF edge_triggered_Dff IS
BEGIN
   state_change: PROCESS (clk, clr) IS
   BEGIN
         IF clr='1' THEN
               q<='0' AFTER 2ns;
         ELSIF clk'EVENT and clk='1' THEN
               q<=d AFTER 2ns;
         end if;
   END PROCESS state_change;
END ARCHITECTURE asyncCLR;

ARCHITECTURE syncCLR OF edge_triggered_Dff IS
BEGIN
   state_change: PROCESS (clk, clr) IS
   BEGIN
         IF clk'event and clk='1' THEN
               IF clr='1' THEN
                     q<='0' AFTER 2ns;
               ELSE
                     Q<=D AFTER 2ns;
               end if;
         end if;
   END PROCESS state_change;
END ARCHITECTURE syncCLR;
```

Primer 5. Multiplekser sa ekskluzivnom selekcijom

```vhdl
-------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--------------------------------------------------
ENTITY muxEx IS
PORT (
   a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
   sel: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
   c: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END muxEx;
--------------------------------------------------
ARCHITECTURE example OF muxEx IS
BEGIN
   PROCESS (a, b, sel)
   BEGIN
         IF (sel="00") THEN
               c <= "00000000";
         ELSIF (sel="01") THEN
               c <= a;
         ELSIF (sel="10") THEN
               c <= b;
         ELSE
               c <= (OTHERS => 'Z');
         END IF;
   END PROCESS;
END ARCHITECTURE example;
```

Primer 6. Jednocifreni BCD brojač

```
ENTITY counter_ent IS
   PORT (clr : IN BIT;
         clk : IN BIT;
         q   : OUT BIT_VECTOR(3 DOWNTO 0));
END ENTITY counter_ent;
 -------------------------------------------
ARCHITECTURE counter_arch OF counter_ent IS
   VARIABLE q_int : BIT_VECTOR(3 DOWNTO 0);
   VARIABLE cq : BIT;
BEGIN
   PROCESS (clr, clk)
   BEGIN
               IF clr='1' THEN
                 q_int := "0000";
                 cq := '0';
               ELSIF clk'event and clk='1' THEN
                 cq := not cq;
                 IF cq='1' THEN
                     CASE q_int IS
                       WHEN "0000" => q_int <= "0001";
                       WHEN "0001" => q_int <= "0010";
                       WHEN "0010" => q_int <= "0011";
                       WHEN "0011" => q_int <= "0100";
                       WHEN "0100" => q_int <= "0101";
                       WHEN "0101" => q_int <= "0110";
                       WHEN "0110" => q_int <= "0111";
                       WHEN "0111" => q_int <= "1000";
                       WHEN "1000" => q_int <= "1001";
                       WHEN OTHERS => q_int <= "0000";
               END CASE;
                 END IF;
               END IF;
   END PROCESS;
   q <= q_int;
END counter_arch;
```

Primer 7. Brojač osnove 16

```
ENTITY counter IS
   PORT (clk, reset: IN bit;
               count : OUT natural);
END ENTITY counter;
-------
ARCHITECTURE behavior OF counter IS
BEGIN
   incrementer: PROCESS IS
         VARIABLE count_value : natural := 0;
   BEGIN
         count <= count_value;
         LOOP
               LOOP
                     WAIT UNTIL clk = '1' or reset = '1';
                     EXIT WHEN reset = '1';
                     count_value := (count_value + 1) mod 16;
                     count <= count_value;
               END LOOP;
               count_value := 0;
```

```
                count <= count_value;
                WAIT UNTIL reset = '0';
          END LOOP;
    END PROCESS incrementer;
END ARCHITECTURE behavior;
```

Primer 8. Registar s paralelnim upisom i serijskim izlazom

```
ENTITY parallel_to_serial IS
   GENERIC (n : integer := 8);
   PORT (wr,clk: IN std_logic;
          d_in: IN std_logic_vector(n-1 DOWNTO 0);
            d_out: OUT std_logic);
END ENTITY parallel_to_serial;
ARCHITECTURE beh  OF parallel_to_serial IS
BEGIN
   PROCESS IS
         VARIABLE int_storage: std_logic_vector(n-1 DOWNTO 0);
   BEGIN
         WAIT UNTIL wr='1';
         int_storage:=d_in;
         FOR i IN n-1 DOWNTO 0 LOOP
               WAIT UNTIL clk'event and clk='1';
               d_out<=int_storage(i);
         END LOOP;
         WAIT UNTIL clk'event and clk='1';
         d_out<='Z';
   END PROCESS;
END ARCHITECTURE beh;
-------
ENTITY parallel_to_serial_tb  IS
   GENERIC(width : integer := 4);
END ;

ARCHITECTURE parallel_to_serial_tb_arch OF parallel_to_serial_tb IS
  SIGNAL wr    :  std_logic  ;
  SIGNAL d_in    :  std_logic_vector (width - 1 downto 0)   ;
  SIGNAL clk    :  std_logic  := '0';
  SIGNAL d_out    :  std_logic  ;

BEGIN
  DUT  : ENTITY work.parallel_to_serial(beh)
    GENERIC MAP (
      n  => width    )
    PORT MAP (
      wr   => wr  ,
      d_in    => d_in   ,
      clk    => clk  ,
      d_out    => d_out    ) ;


   clk <= not clk after 50 ns;

   stimuli: process
   BEGIN
       d_in <= "0101";
       wr<='1';
       WAIT FOR 50 ns;
       wr<='0';
```

```
        WAIT FOR 600 ns;
        d_in <= "1101";
        wr<='1';
        WAIT FOR 100 ns;
        d_in<="0000";
        WAIT FOR 500 ns;
    END PROCESS stimuli;
END ;
```

Primer 9. Sinhroni 8b brojač sa dozvolom brojanja

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY counter8 IS
   PORT (
     clk: IN STD_LOGIC;
     reset: IN STD_LOGIC;
     ce, load, dir: IN STD_LOGIC;
     din: IN INTEGER RANGE 0 TO 255;
     count: OUT INTEGER RANGE 0 TO 255

   );
END counter8;

ARCHITECTURE counter8_arch OF counter8 IS
BEGIN

PROCESS (clk, reset)
VARIABLE counter: INTEGER RANGE 0 TO 255;
BEGIN
  IF reset='1' THEN
     counter := 0;
  ELSIF clk='1' and clk'EVENT THEN
     IF load='1' THEN
        counter := din;
     ELSE
        IF ce='1' THEN
           IF dir='1' THEN
            IF counter =255 THEN
                counter := 0;
             ELSE
                counter := counter + 1;
             END IF;
           ELSE
             IF counter =0 THEN
                counter := 255;
             ELSE
                counter := counter - 1;
             END IF;
           END IF;
        END IF;
     END IF;
  END IF;
  count <= counter;
END PROCESS;

END counter8_arch;
```