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 2 -

Primer 10. Memorija, sinhrona

```
01 LIBRARY IEEE;
02 USE IEEE.STD_LOGIC_1164.ALL;
03 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
04 ENTITY Memorija IS
05 PORT ( WE : IN STD_LOGIC;
06    clk : in STD_LOGIC;
07    addr : in STD_LOGIC_VECTOR (7 downto 0);
08    data : in STD_LOGIC_VECTOR (7 downto 0);
09    Q : out STD_LOGIC_VECTOR (7 downto 0)
10    );
11 end Memorija;
12 architecture Behavioral of Memorija is
13    type ram_mem_type is array (255 downto 0) of STD_LOGIC_VECTOR(7 downto
0);
14    signal rammem : ram_mem_type;
15 begin
16
```

```
process (clk)
variable addrtemp: integer range 255 downto 0;
begin
   if (clk'EVENT and clk = '0') then
         addrtemp := CONV_INTEGER(addr);
         if (WE= '1') then
               rammem(addrtemp) <=data;
         end if ;
         Q<= rammem(addrtemp);
   end if ;
end process;
end architecture;
```

Primer 11. Brojač vodećih nula u ulaznom podatku

```
 LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;

 ENTITY LeadingZeros IS
 GENERIC( n : INTEGER := 8);
 PORT ( data: IN STD_LOGIC_VECTOR (n-1 DOWNTO 0);
 zeros: OUT INTEGER RANGE 0 TO N);
 END LeadingZeros;
 ARCHITECTURE behavior OF LeadingZeros IS
 BEGIN
   PROCESS (data)
         VARIABLE count: INTEGER RANGE 0 TO n;
   BEGIN
         count := 0;
         FOR i IN data'RANGE LOOP
         CASE data(i) IS
                     WHEN '0' => count := count + 1;
                     WHEN OTHERS => EXIT;
               END CASE;
         END LOOP;
         zeros <= count;
   END PROCESS;
END behavior;
```

Primer 12. Kolo za kašnjenje

```
ENTITY shift_reg IS
   PORT (clk : IN bit;
         din : IN integer;
         dout   : OUT integer);
END ENTITY shift_reg;
 -------------------------------------------------------
ARCHITECTURE arch OF shift_reg IS
   type int_array is array (0 to 3) of integer;
   SIGNAL d: int_array;
BEGIN
   PROCESS (clk)
   BEGIN
```

```
13          if (clk'event and clk='1') then
14                d<= d(1 to 3) & din;
15          end if;
16
17    END PROCESS;
18    dout <= d(0);
19 END arch;
20 ------------------
21 entity shift_reg_tb is
22 end shift_reg_tb;
23 architecture shift_reg_tb_arch of shift_reg_tb is
24
25    SIGNAL clk: bit :='0';
26    signal din, dout : integer;
27
28 begin
29 UUT: entity work.shift_reg(arch)
30    port map
31    (
32          clk => clk,
33          din => din,
34          dout => dout
35    );
36 PROCESS (CLK)
37 BEGIN
38 clk<=not clk after 50 ps;
39 END PROCESS;
40 stimuli: process is
41 begin
42    din<=1, 2 after 100 ps, 3 after 200 ps, 4 after 300 ps, 5 after 400 ps;
43    wait for 500 ps;
44 end process stimuli;
45 end shift_reg_tb_arch;
```

Primer 13. Adresni dekoder, izlazni port aktivno niskog nivoa

```
01 library ieee;
02 use ieee.std_logic_1164.all;
03 use ieee.numeric_std.all;
04 ENTITY test IS
05    generic (n : positive:=2);
06    PORT (EN : IN std_logic;
07          address : IN  unsigned(n-1 downto 0);
08          decoded_address : OUT std_logic_vector(2**n-1 downto 0)
09          );
10 END ENTITY test;
11  ----------------------------------------------------
12 ARCHITECTURE arch OF test IS
13 BEGIN
14    PROCESS (EN, address)
15          variable internal: std_logic_vector(2**n-1 downto 0);
16          variable addr: natural range 0 to 2**n-1;
17    BEGIN
18          addr := to_integer(address);
19
```

```
20          internal:= (others=>'1');
21          if (EN='1') then
22                internal(addr):='0';
23          end if;
24          decoded_address <= internal;
25    END PROCESS;
26 END arch;
27 ------------------
28 library ieee;
29 use ieee.std_logic_1164.all;
30 use ieee.numeric_std.all;
31 entity test_tb is
32    generic( n: positive := 4);
33 end test_tb;
34 architecture test_tb_arch of test_tb is
35    SIGNAL EN: std_logic;
36    signal address : unsigned(n-1 downto 0);
37    signal decoded_address : std_logic_vector(2**n-1 downto 0);
38 begin
39 UUT: entity work.test(arch)
40    generic map(n=>n)
41    port map
42    (
43          EN => EN,
44          address => address,
45          decoded_address => decoded_address
46    );
47 stimuli: process is
48 begin
49    EN <= '0', '1' after 100 ps;
50    for i in integer range 0 to 2**n-1 loop
51          wait for 100 ps;
52          address<=to_unsigned(i, n);
53    end loop;
54    wait for 100 ps;
55 end process stimuli;
56 end test_tb_arch;
```

Primer 14. Carry-ripple sabirač izveden konkurentnim klauzulama dodele vrednosti signalu.

```
01 LIBRARY ieee;
02 USE ieee.std_logic_1164.all;
03 ENTITY carry_ripple_adder IS
04    GENERIC (n: INTEGER := 4);
05  PORT ( a, b: IN STD_LOGIC_VECTOR (n-1 DOWNTO 0);
06               cin: IN STD_LOGIC;
07               s: OUT STD_LOGIC_VECTOR (n-1 DOWNTO 0);
08               cout: OUT STD_LOGIC);
09  END carry_ripple_adder;
10  ARCHITECTURE w_generate OF carry_ripple_adder IS
11  SIGNAL c_int: STD_LOGIC_VECTOR (n DOWNTO 0);
12  BEGIN
13    c_int(0) <= cin;
```

```
14    G1: FOR i IN 0 TO n-1 GENERATE
15          s(i) <= a(i) XOR b(i) XOR c_int(i);
16          c_int(i+1) <= (a(i) AND b(i)) OR (a(i) AND c_int(i)) OR (b(i) AND
      c_int(i));
17  END GENERATE;
18     cout <= c_int(n);
19  END ARCHITECTURE w_generate;
```

Primer 15. Registar sa dozvolom izlaza, sačinjen od Dff-ova i trostatičkih bafera.

```
01 LIBRARY ieee;
02 USE ieee.std_logic_1164.ALL;
03 ENTITY D_flipflop IS
04    PORT (clk, d: IN std_logic; q: OUT std_logic);
05 END ENTITY D_flipflop;
06 ARCHITECTURE simple OF D_flipflop IS
07 BEGIN
08    PROCESS(clk)
09    BEGIN
10          IF clk'event and clk='1' THEN
11                q<=d;
12          END IF;
13    END PROCESS;
14
15 END ARCHITECTURE simple;
16 --------
17
18 LIBRARY  ieee;
19 USE ieee.std_logic_1164.ALL;
20 -- komponente koje se koriste:
21 ENTITY tristate_buffer IS
22    PORT (a, en: IN STD_LOGIC; y: OUT STD_LOGIC);
23 END ENTITY tristate_buffer;
24 ARCHITECTURE beh OF tristate_buffer IS
25 BEGIN
26    y<=a WHEN en='1' ELSE 'Z';
27 END ARCHITECTURE beh;
28 --------
29
30 LIBRARY ieee;
31 USE ieee.std_logic_1164.ALL;
32 ENTITY register_tristate IS
33    GENERIC (width: positive);
34    PORT (clock: IN std_logic;
35          out_enable: IN std_logic;
36          data_in: IN std_logic_vector (0 TO width-1);
37          data_out: OUT std_logic_vector (0 TO width-1));
38 END ENTITY register_tristate;
39 ARCHITECTURE cell_level OF register_tristate IS
40 BEGIN
41    cell_array: FOR bit_index IN 0 TO width-1 GENERATE
42          SIGNAL data_unbuffered: std_logic;
43    BEGIN
```

```
44          cell_storage: ENTITY work.D_flipflop(simple)
45                PORT MAP (clk=>clock, d=>data_in(bit_index),
46                      q=>data_unbuffered);
47          cell_buffer: ENTITY work.tristate_buffer(beh)
48                PORT MAP (a=>data_unbuffered, en=>out_enable,
49                      y=>data_out(bit_index));
50    END GENERATE cell_array;
51 END ARCHITECTURE cell_level;
```

Primer 16. Pomerački registar sa serijskim ulazom i paralelnim izlazom; koristi Dff iz prethodnog primera.

```
01 LIBRARY  ieee;
02 USE ieee.std_logic_1164.ALL;
03 ENTITY shift_reg IS
04    GENERIC(n: natural := 4);
05    PORT (clk: IN std_logic;
06          serial_data_in: IN std_logic;
07          parallel_data: INOUT std_logic_vector(n-1 DOWNTO 0)
08          );
09 END ENTITY shift_reg;
10 ARCHITECTURE cell_level OF shift_reg IS
11 BEGIN
12    reg_array: FOR index IN parallel_data'RANGE GENERATE
13    BEGIN
14          first_cell: IF index= parallel_data'left GENERATE
15          BEGIN
16                cell: ENTITY work.D_flipflop(simple)
17                PORT MAP (clk=>clk,
18                      d=>serial_data_in,
19                      q=>parallel_data(index));
20          END GENERATE first_cell;
21          non_first_cell: IF index/= parallel_data'left GENERATE
22          BEGIN
23                cell: ENTITY work.D_flipflop(simple)
24                PORT MAP (clk=>clk,
25                      d=>parallel_data(index+1),
26                      q=>parallel_data(index));
27          END GENERATE non_first_cell;
28    END GENERATE reg_array;
29 END ARCHITECTURE cell_level;
```