Deo 1: Modelovanje kombinacione logike u VHDL-u

1.1 Entiteti i arhitekture

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity and2 is
      port(x,y:in bit; z:out bit);
end entity and2;

architecture ex1 of and2 is
begin
      z<=x and y;
end architecture ex1;
```

1.2 Identifikatori, komentari i razmaci

1.3 Netliste

--- Z=(not A and B) or (A and C)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity comb_function is
      port(a,b,c:in bit; z:out bit);
end entity comb_function;

architecture expression of comb_function is
begin
      z<=(not a and b) or (a and c);
end architecture expression;
---------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity or2 is
      port(x,y:in bit; z:out bit);
end entity or2;

architecture ex1 of or2 is
begin
      z<=x or y;
end architecture ex1;
-----------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity not1 is
       port( x : in bit; z : out bit);
end entity not1;

architecture ex1 of not1 is
begin
```

```
      z<=not x;
end architecture ex1;
-------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity comb_function is
      port(a,b,c:in bit; z:out bit);
end entity comb_function;

architecture netlist of comb_function is

      component and2 is
            port(x,y:in bit; z:out bit);
      end  component and2;
      component or2 is
            port(x,y:in bit; z:out bit);
      end component or2;
      component not1 is
            port(x:in bit; z:out bit);
      end component not1;

      signal p,q,r :bit;

begin
      lab1: not1 port map(a,p);
      lab2: and2 port map (p,b,q);
      lab3: and2 port map(a,c,r);
      lab4: or2 port map (q,r,z);
end architecture netlist;
----------------------------------------

lab2: and2 port map (z=>q, x=>p, y=>b);

architecture direct of comb_function is
      signal p,q,r :bit;
begin
      lab1: entity work.not1(ex1) port map(a,p);
      lab2: entity work.and2(ex1) port map (p,b,q);
      lab3: entity work.and2(ex1) port map(a,c,r);
      lab4: entity work.or2(ex1) port map (q,r,z);
end architecture direct;
```

1.4 Dodeljivanje vrednosti signalima

```
z<=x and y;

z<=x;

z<=not (( x and y) or (a and b));

z<=x after 4 ns;

z<= transport x after 4 ns;

z<=x and y after 5 ns;

z<= transport not ((x and y) or (a and b)) after 5 ns;
```

```
z<= x after 4 ns, not x after 8 ns;
```

1.5 Generici

```
z<=x and y after 5 ns;
z<=x and y after delay;
-------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity and2 is
      generic(delay:time);
      port(x,y:in bit; z:out bit);
end entity and2;

architecture ex2 of and2 is
begin
      z<=x and y after delay;
end architecture ex2;
-------------------------------

lab1: and2 generic map (5 ns) port map (p,b,q);

lab2: and2 generic map (delay=> 5 ns) port map(z=>q, x=>p, y=>b);

generic(delay:time:=5ns);

lab3: and2 port map (p,b,q);

lab4: and2 generic map (3 ns) port map (p,b,q);

lab5: and2 generic map (open) port map (p,b,q);
```

1.6 Konstantni i otvoreni portovi

```
-------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity universal is
      port(x,y,invert:in bit; a,o:out bit);
end entity universal;


architecture univ of universal is
begin
      a<=(y and (x xor invert)) or (invert and not y);
      o<=(not x and (y xor invert)) or (x and not invert);
end architecture univ;
-------------------------------

u0: universal port map (x,y,'0',a,open);

entity universal is
      port( x,y:in bit;
            invert:in bit:='0';
```

```
            a,o:out bit);
end entity universal;

u0: universal port map (x,y,open,a,open);
```

1.7 Testbenčevi

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

entity and2_tb is
end entity and2_tb;

architecture TB_ARCHITECTURE_1 of and2_tb is
      component and2
            port( x,y : in BIT; z : out BIT );
      end component;
      signal a,b,c : BIT;
begin

      lab1 : and2 port map (x => a,y => b,z => c);
      a<='0','1' after 100 ns;
      b<='0','1' after 150 ns;

end architecture TB_ARCHITECTURE_1;
-----------------------------------
```

1.8 Konfiguracije

```
-----------------------------
architecture ex3 of and2 is
    signal xy : bit_vector(0 to 1);
begin
    xy<=x&y;
    with xy select
        z<='1' when "11",
           '0' when others;
end architecture ex3;
-----------------------------

lab1: entity work.not1(ex1) port map (a,p);

for lab1:and2 use entity work.and2(ex1);

------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity and2_tb is
end entity and2_tb;

architecture TB_ARCHITECTURE_2 of and2_tb is
      component and2
            port( x,y : in BIT; z : out BIT );
      end component;
      for all: and2 use entity work.and2(ex1);
```

```
      signal a,b,c : BIT;
begin

      lab1 : and2 port map (x => a,y => b,z => c);
      a<='0','1' after 100 ns;
      b<='0','1' after 150 ns;

end architecture TB_ARCHITECTURE_2;


--------------------------------
configuration Konfiguracija1 of and2_tb is
      for TB_ARCHITECTURE_1
            for lab1 : and2
                  use entity work.and2(ex1);
            end for;
      end for;
end Konfiguracija1;

-------------------------------------------------------



library ieee;
use ieee.std_logic_1164.all;

entity and2_tb is
end entity and2_tb;

architecture TB_ARCHITECTURE_3 of and2_tb is
      component and2
            generic(dly:time);
            port( in1,in2 : in BIT; out1 : out BIT );
      end component and2;
      signal a,b,c : BIT;
begin

      lab1 : and2 generic map(6 ns) port map (a,b,c);
      a<='0','1' after 100 ns;
      b<='0','1' after 150 ns;

end architecture TB_ARCHITECTURE_3;

configuration Konfiguracija2 of and2_tb is
      for TB_ARCHITECTURE_3
            for lab1 : and2
                  use entity work.and2(ex2)
                  generic map(delay=>dly)
                  port map (x=>in1,y=>in2,z=>out1);
            end for;
      end for;
end Konfiguracija2;
```

Deo 2: Kombinacioni blokovi

2.1 *Three-state* baferi

2.1.1 Višeznačna logika

```
type tri is ('0','1','Z');

signal a,b,c : tri;

a<='0' and '1';
b<=a or c after 5 ns;

function "and" (Left,Right:tri) return tri is
type tri_array is array(tri,tri) of tri;
constant and_table : tri_array:=(('0','0','0'),
                                 ('0','1','1'),
                                 ('0','1','1'));
begin
 return and_table(Left, Right);
end function "and";
```

2.1.2 *Standard logic package*

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

2.1.2 *When ... else* naredba

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity three_state is
      port(a,enable:in std_ulogic; z:out std_ulogic);
end entity three_state;

architecture when_else of three_state is
begin
      z<=a  when enable = '1' else 'Z';
end architecture when_else;

architecture after_when_else of three_state is
begin
      z<=a  after 5 ns when enable = '1' else 'Z';
end architecture after_when_else;
```

2.2 Dekoderi

2.2.1 2 u 4 dekoder

```
type std_ulogic_vector is array ( natural range <>) of std_ulogic;
type std_logic_vector is array ( natural range <>) of std_logic;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity decoder is
      port( a : in  std_ulogic_vector(1 downto 0);
            z : out std_ulogic_vector(3 downto 0));
end entity decoder;

architecture when_else of decoder is
begin
      z<=”0001” when a = “00” else
         ”0010” when a = “01” else
         ”0100” when a = “10” else
         ”1000” when a = “11” else
         “XXXX”;
end architecture when_else;
-------------------------------------

      z<=”0001” when a = “00” else
         ”0010” when a = “01” else
         ”0100” when a = “10” else
         ”1000” when a = “11”;

      z<=”0001” when a = “00” else
         ”0010” when a = “01” else
         ”0100” when a = “10” else
         ”1000” when a = “11” else
         unaffected;

      z<=”0001” when a = “00” else
         ”0010” when a = “01” else
         ”0100” when a = “10” else
         ”1000” when a = “11” else
         z;

z<=”0001” when (a = “00” and (en=’1’ or en2=’0)) else …
```

2.2.2 *With ... select* naredba

```
architecture with_select of decoder is
begin
   with a select
      z<=”0001” when  “00” ,
         ”0010” when  “01” ,
         ”0100” when  “10” ,
         ”1000” when  “11” ,
         “XXXX” when others ;
end architecture with_select;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity seven_seg is
      port( a : in  std_ulogic_vector(3 downto 0);
            z : out std_ulogic_vector(6 downto 0));
end entity seven_seg;

architecture with_select of decoder is
begin
   with a select
      z<=”1110111” when  “0000” ,
         ”0010010” when  “0001” ,
         ”1011101” when  “0010” ,
         ”1011011” when  “0011” ,
         ”0111010” when  “0100” ,
         ”1101011” when  “0101” ,
         ”1101111” when  “0110” ,
         ”1010010” when  “0111” ,
         ”1111111” when  “1000” ,
         ”1111011” when  “1001” ,
         ”1101101” when  “1010”|“1011”|”1100”|“1101”|”1110”|”1111”,
         “0000000” when others;
end architecture with_select;
-----------------------------------------
”1101101” when  “1010” to ”1111”,
```

2.2.3 n u 2^n dekoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity decoder is
      generic( n:positive)
      port( a : in  std_ulogic_vector(n-1 downto 0);
            z : out std_ulogic_vector(2**n-1 downto 0));
end entity decoder;

constant z_out :bit_vector(2**n-1 downto 0):=(0=>’1’,others=>’0’);

sll, sla, rol, ror srl, sra

x<=unsigned(y)
y<=std_ulogic_vector(x);

--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;

entity decoder is
      generic( n:positive)
      port( a : in  std_ulogic_vector(n-1 downto 0);
            z : out std_ulogic_vector(2**n-1 downto 0));
end entity decoder;
```

```
architecture rotate of decoder is
constant z_out :bit_vector(2**n-1 downto 0):=(0=>'1',others=>'0');
begin
      z<=to_StdULogicVector(z_out sll to_integer(unsigned(a)));
end architecture rotate;
-----------------------------------------
```

2.3 4 u 1 multiplekser

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux is
      port( a,b,c,d : in  std_ulogic;
            s: in std_ulogic_vector(1 downto 0);
            y: out std_ulogic);
end entity mux;

architecture with_select of mux is
begin
  with s select
      y<=a when "00",
         b when "01",
         c when "10",
         d when "11",
         'X' when others;
end architecture with_select;
-----------------------------------

architecture when_else of mux is
begin
      y<=a when s = "00" else
         b when s = "01" else
         c when s = "10" else
         d when s = "11" else
         'X';
end architecture when_else;
-----------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux is
      port( a,b,c,d : in  std_ulogic;
            s: in std_ulogic_vector(1 downto 0);
            y: out std_ulogic);
end entity mux;

architecture three_state of mux is
begin
      y<= a when s = "00" else 'Z';
      y<= b when s = "01" else 'Z';
      y<= c when s = "10" else 'Z';
      y<= d when s = "11" else 'Z';
```

```
end architecture three_state;
```

2.4 Enkoder prioriteta

2.4.1 *Don't Cares*

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity priority is
      port( a :  in  std_ulogic_vector(3 downto 0);
            y :  out std_ulogic_vector(1 downto 0);
          valid: out std_ulogic);
end entity priority;

architecture dont_care of priority is
begin
      with a select
      y<=”00” when “0001”,
         “01” when “001-”,
         “10” when “01--”,
         “11” when “1---”,
         “00” when others;
      valid<=’1’ when
         (a(0)=’1’ or a(1)=’1’ or a(2)=’1’ or a(3)=’1’) else ‘0’;
end architecture dont_care;


architecture ordered of priority is
begin
      y<=”00” when a(3)=’1’ else
         “01” when a(2)=’1’ else
         “10” when a(1)=’1’ else
         “11” when a(0)=’1’ else
         “00” ;
      valid<= ’1’ when
         (a(0)=’1’ or a(1)=’1’ or a(2)=’1’ or a(3)=’1’) else ‘0’;
end architecture ordered;

---------------------------------------
use numeric_std.all;
architecture match of priority is
begin
      y<=”00” when std_match(a,”0001”) else
         “01” when std_match(a,”001-”)  else
         “10” when std_match(a,”01--”) else
         “11” when std_match(a,”1---”) else
         “00” ;
      valid<= ’1’ when
         (a(0)=’1’ or a(1)=’1’ or a(2)=’1’ or a(3)=’1’) else ‘0’;
end architecture match;
```

2.4.2 Sekvencijalni VHDL

```
architecture sequential of priority is
begin
 process(a) is
 begin
  if a(3)=’1’ then
    y<=”11”;
    valid<=’1’;
  elsif a(2)=’1’ then
    y<=”10”;
    valid<=’1’;
  elsif a(1)=’1’ then
    y<=”01”;
    valid<=’1’;
  elsif a(0)=’1’ then
    y<=”00”;
    valid<=’1’;
  else
    y<=”00”;
    valid<=’0’;
  end if;
 end process;
end architecture sequential;

-----------------------------------

architecture sequential2 of priority is
begin
 process(a) is
 begin
  valid<=’1’;
  if a(3)=’1’ then
    y<=”11”;
  elsif a(2)=’1’ then
    y<=”10”;
  elsif a(1)=’1’ then
    y<=”01”;
  elsif a(0)=’1’ then
    y<=”00”;
  else
    y<=”00”;
    valid<=’0’;
  end if;
 end process;
end architecture sequential2;

---------------------------------------
```

2.5 Sabirači

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

entity FA1 is
      port (
              a: in STD_LOGIC; -- Inputs
              b: in STD_LOGIC;
```

```
              ci: in STD_LOGIC; -- Carry input
              co: out STD_LOGIC; -- Carry output
              z: out STD_LOGIC -- Sum
      );
end FA1;

architecture Concurent of FA1 is
begin

  z <= a xor b xor ci;
  co <= (a and b) or (a and ci) or (b and ci);

end Concurent;


--------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity adder1 is
      generic(N:natural:=24); -- Adder width

      port( in1: in std_logic_vector(N-1 downto 0); -- Inputs
            in2: in std_logic_vector(N-1 downto 0);
            cin: in std_logic; -- Carry input
            sum: out std_logic_vector(N-1 downto 0) -- Sum
            );
end adder1;

architecture Structural of adder1 is
      component FA1 is
            port( a,b,ci:in std_logic;  co,z:out std_logic);
      end component;
      signal c : std_logic_vector(N-1 downto 0);
begin

      elements : for i in N-1 downto 0 generate
            LSB: if i = 0 generate
                  fad16: FA1 port map(a=>in1(0),b=>in2(0),
ci=>cin,co=>c(0),z=>sum(0));
                  end generate;
            MSBs: if i > 0 generate
                  fad17: FA1 port map(a=>in1(i),b=>in2(i),
ci=>c(i-1),co=>c(i),z=>sum(i));
                  end generate;
      end generate;

end Structural;
```

2.5 Parity Checker

```
Loop
While condition loop
```

For *identifier* in *range* loop

For I in 0 to n-1 loop

For I in a'range loop

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

entity parity is
  port(a :in std_ulogic_vector;
       y:out std_ulogic);
end entity parity;

architecture iterative of parity is
begin
  process(a) is
    variable even:std_ulogic;
  begin
    even:='0';
       for I in a'range loop
          if a(i)='1' then even:=not even;
          end if;
       end loop;
    y<=even;
  end process;
end architecture iterative;
```
