FAKULTET ELEKTROTEHNIKE
I RAČUNARSTVA

Zavod za elektroničke sustave
i obradbu informacija

ELEMENTI I PRIMJENA JEZIKA VHDL

Riješeni primjeri

Marko Butorac
Mladen Vučić

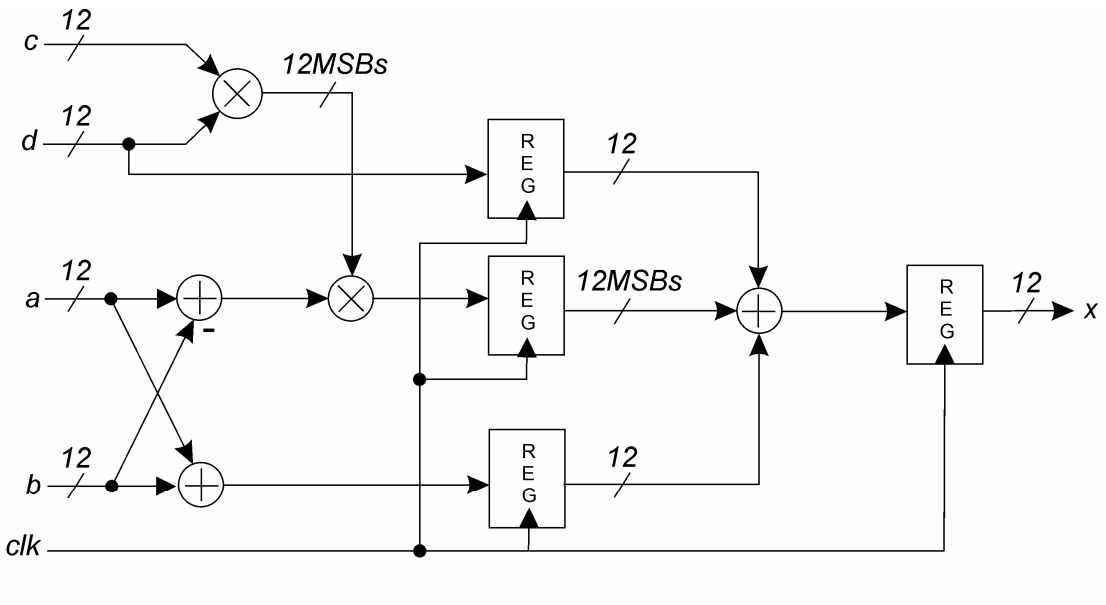
Riješeni primjeri

1. Napisati funkciju čiji je ulazni argument klase konstanta tipa `bit_vector`, a izlaz tipa `natural`. Funkcija pretvara `bit_vector` u `natural`. Indeksi ulaznog vektora su padajući.

Rješenje

```
function bv_to_natural(bv : in bit_vector) return natural is
   variable vrijednost  : natural := 0;
begin
   for index in bv'left downto bv'right loop
      if bv(index) = '1' then
         vrijednost := vrijednost + 2**index;
      end if;
   end loop;
   return vrijednost;
end function;
```

2. Realizirati sklop prikazan slikom. Ulazi u sklop su **a**, **b**, **c**, **d** i **clk**, a izlaz **x**. Pored ovih signala potrebno je predvidjeti signal za sinkroni reset registara. Pretpostaviti da signali **a**, **b**, **c** i **d** sadrže vrijednosti u dvojnom komplementu. (Napomena: *MSBs* označava *Most Significant Bits* tj. najznačajnije bitove.)

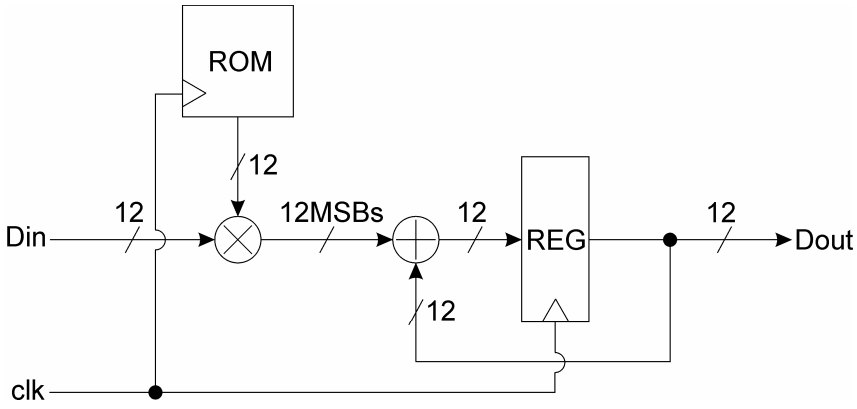


Rješenje

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity zad2 is
   port  (
         clk   : in std_logic;
         reset : in std_logic;

         a     : in std_logic_vector(11 downto 0);
         b     : in std_logic_vector(11 downto 0);
         c     : in std_logic_vector(11 downto 0);
         d     : in std_logic_vector(11 downto 0);

         x     : out std_logic_vector(11 downto 0)
         );
end zad2;

architecture rtl of zad2 is

   -- local signals
   signal mul_c_d          : std_logic_vector(23 downto 0);
   signal mul_c_d_short    : std_logic_vector(11 downto 0);
   signal sum_a_b          : std_logic_vector(11 downto 0);
   signal sub_a_b          : std_logic_vector(11 downto 0);
   signal mul2             : std_logic_vector(23 downto 0);
   signal mul2_short       : std_logic_vector(11 downto 0);
   signal reg1, reg2, reg3 : std_logic_vector(11 downto 0);
   signal sum_regs         : std_logic_vector(11 downto 0);
   signal reg_out          : std_logic_vector(11 downto 0);

begin
   -- multiplying c x d
   mul_c_d <= c * d;
   mul_c_d_short <= mul_c_d(23 downto 12);

   -- sum and subtract of a and b
   sum_a_b <= a + b;
   sub_a_b <= a - b;

   -- multiplying signals sub_a_b and mul_c_d_short
   mul2 <= sub_a_b * mul_c_d_short;
   mul2_short <= mul2(23 downto 12);
```

```
    -- register for signal d, mul2_short, sum_a_b
    process(clk)
    begin
       if rising_edge(clk) then
          if reset = '1' then
             reg1 <= (others => '0');
             reg2 <= (others => '0');
             reg3 <= (others => '0');
          else
             reg1 <= d;
             reg2 <= mul2_short;
             reg3 <= sum_a_b;
          end if;
       end if;
    end process;

    -- sum of signals reg1, reg2, reg3
    sum_regs <= reg1 + reg2 + reg3;

    -- output register
    process(clk)
    begin
       if rising_edge(clk) then
          if reset = '1' then
             reg_out <= (others => '0');
          else
             reg_out <= sum_regs;
          end if;
       end if;
    end process;

    -- output mapping
    x <= reg_out;

end rtl;
```

3. Realizirati sklop prikazan slikom. Ulazi u sklop su 12-bitni podatak **Din**, **clk** i sinkroni reset **rst**, a izlaz je 12-bitni podatak **Dout**. ROM memoriju sa 128 lokacija realizirati korištenjem blok RAM-a. Blok RAM realizirati korištenjem slijednog opisa. Generator adresa za memoriju realizirati unutar sklopa korištenjem sinkronog brojila.



Rješenje

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity zad3 is
   port  (
         clk   : in std_logic;
         reset : in std_logic;

         Din   : in std_logic_vector(11 downto 0);
         Dout  : out std_logic_vector(11 downto 0)
         );
end zad3;

architecture RTL of zad3 is
   -- local signals
   signal mul_out           : std_logic_vector(23 downto 0);
   signal sum_out, reg_out  : std_logic_vector(11 downto 0);
   signal mem_location      : std_logic_vector(6 downto 0);

   -- block RAM
   type BRAM is array (0 to 127)
                of std_logic_vector(11 downto 0);
   signal rom   : BRAM;
   signal we    : std_logic;
   signal data_in, data_out : std_logic_vector(11 downto 0)
                            := (others => '0');
```

```
begin
   -- ROM memory
   process (clk) is
   begin
      if rising_edge(clk) then
         if (we = '1') then
            rom(conv_integer(mem_location)) <= data_in;
         else
            data_out <= rom(conv_integer(mem_location));
         end if;
      end if;
   end process;

   -- address generator
   process (clk) is
   begin
      if rising_edge(clk) then
         if (reset = '1') then
            mem_location <= (others => '0');
         else
            mem_location <= mem_location + 1;
         end if;
      end if;
   end process;

   -- multiplier
   mul_out <= Din * data_out;

   -- sumator
   sum_out <= mul_out(23 downto 12) + reg_out;

   -- register
   process (clk) is
   begin
      if rising_edge(clk) then
         if (reset = '1') then
            reg_out <= (others => '0');
         else
            reg_out <= sum_out;
         end if;
      end if;
   end process;

   -- output mapping
   Dout <= reg_out;

end RTL;
```

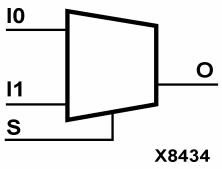
4. Korištenjem isključivo komponente iz biblioteke čiji je opis dan na slijedećoj stranici realizirati 4/1 multipleksor. Ulazi u sklop su 4 1-bitna signala, **a**, **b**, **c**, **d**, 2-bitni upravljački signal, **s**, a izlazni signal je **y**.

MUXF6

2-to-1 Lookup Table Multiplexer with General Output

Architectures Supported

MUXF6	
Spartan-II, Spartan-IIE	Primitive
Spartan-3	Primitive
Virtex, Virtex-E	Primitive
Virtex-II, Virtex-II Pro, Virtex-II Pro X	Primitive
XC9500, XC9500XV, XC9500XL	No
CoolRunner XPLA3	No
CoolRunner-II	No



MUXF6 provides a multiplexer function in a full Virtex, Virtex-E, Spartan-II, or Spartan-IIE CLB, or one half of a Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, "MUXF6_D" and "MUXF6_L", provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

Inputs			Outputs
S	I0	I1	O
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```
-- Component Declaration for MUXF6 should be placed
-- after architecture statement but before begin keyword

component MUXF6
   port (O  : out STD_ULOGIC;
         I0 : in STD_ULOGIC;
         I1 : in STD_ULOGIC;
         S : in STD_ULOGIC);
end component;
```

Rješenje

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if
-- instantiating any Xilinx primitives in this code.
library UNISIM;
use UNISIM.VComponents.all;

entity zad4 is
   port  (
         a, b, c, d  : in std_logic;
         s           : in std_logic_vector(1 downto 0);
         y           : out std_logic
         );
end zad4;

architecture RTL of zad4 is
   -- local signals
   signal o1, o2     : std_logic;
begin

MUXF6_inst1 : MUXF6
      port map (
         O => o1,
         I0 => a,
         I1 => b,
         S => s(0)
      );

MUXF6_inst2 : MUXF6
      port map (
         O => o2,
         I0 => c,
         I1 => d,
         S => s(0)
      );

MUXF6_inst3 : MUXF6
      port map (
         O => y,
         I0 => o1,
         I1 => o2,
         S => s(1)
      );

end RTL;
```