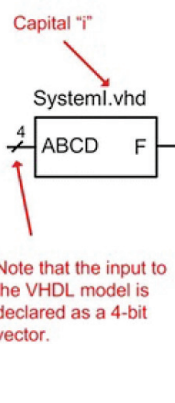


4. Основе VHDL-а (део 2)

Задатак 8.2.1

Написати VHDL модел који имплементира понашање описано 4-улазном таблицом истинитости на слици 8.4. Користити процес и if/then наредбу. Користити типове std_logic и std_logic_vector за сигнале. Декларисати ентитет тако да описује блок дијаграм са слике. Примећује се да има више улазних кодова који дају $F = 0$ него $F = 1$. Да ли ово може да се искористи да VHDL модел буде простији?

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Слика. 8.4 System I функционалност

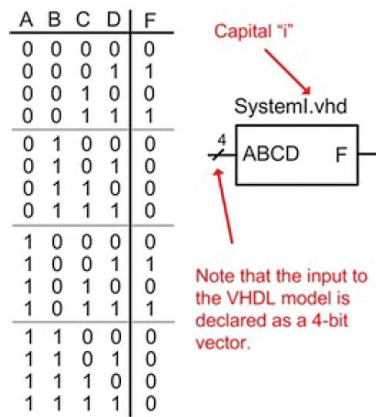
Решење задатка 8.2.1

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity SystemI is
    port (ABCD : in std_logic_vector(3 downto 0);
          F: out std_logic);
end entity;

architecture SystemI_arch of SystemI is
begin
    SystemI_Proc : process (ABCD)
    begin
        if      (ABCD="0001") then F <= '1';
        elsif  (ABCD="0011") then F <= '1';
        elsif  (ABCD="1001") then F <= '1';
        elsif  (ABCD="1011") then F <= '1';
        else F <= '0';
        end if;
    end process;
end architecture;
```

Задатак 8.2.2

Написати VHDL модел који имплементира понашање описано 4-улазном таблицом истинитости на слици 8.4. Користити процес и case наредбу. Користити типове std_logic и std_logic_vector за сигнале. Декларисати ентитет тако да описује блок дијаграм са слике.



Слика. 8.4 System I функционалност

Решење задатка 8.2.2

```

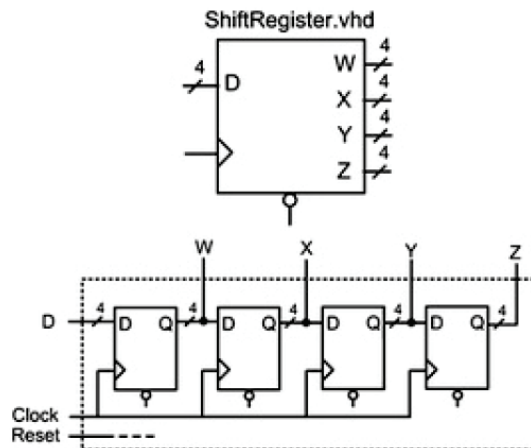
Library IEEE;
use IEEE.std_logic_1164.all;
entity SystemI is
    port (ABCD : in std_logic_vector(3 downto 0);
          F: out std_logic);
end entity;

architecture SystemI_arch of SystemI is
begin
    SystemI_Proc : process (ABCD)
    begin
        case (ABCD) is
            when "0001" | "0011" | "1001" | "1011") => F <= '1';
            when others => F <= '0';
        end case;
    end process;
end architecture;

```

Задатак 8.2.9

Слика 8.8 показује топологију 4-битног померачки регистра имплементирану преко D флип-флопа. Написати VHDL модел који описује функционалност коришћењем једног процеса и секвенцијалну доделу сигнала уместо инстанцирања D флип-флопова. Слика такође приказује блок дијаграм за дефиницију ентитета. Користити типове std_logic и std_logic_vector за сигнале.



Слика. 8.8 Функционалност 4-битног померачког регистра

Решење задатка 8.2.9

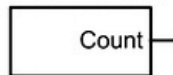
```
Library IEEE;
use IEEE.std_logic_1164.all;
entity ShiftRegister is
    port (D : in std_logic_vector(3 downto 0);
          Clock : in std_logic;
          W : out std_logic_vector(3 downto 0);
          X : out std_logic_vector(3 downto 0);
          Y : out std_logic_vector(3 downto 0);
          Z : out std_logic_vector(3 downto 0));
end entity;

architecture ShiftRegister_arch of ShiftRegister is
begin
    ShiftRegister_Proc : process (Clock)
    begin
        if (rising_edge(Clock)) then
            Z <= Y;
            Y <= X;
            X <= W;
            W <= D;
        End if;
    end process;
end architecture;
```

Задатак 8.2.10

Написати VHDL модел за бројач коришћењем for петљу са излазним типом integer. Слика 8.9 приказује блок дијаграм за дефиницију ентитета. Бројач треба да се инкрементира од 0 до 31 а онда да почне поново. Користити wait наредбу у оквиру процеса да би се бројач ажурира на ваких 10ns. Размотрити коришћење променљиве у петљи која генерише вредност бројача. Напомена: Овај модел се не може синтетизовати.

counter_integer_up.vhd



Слика. 8.8 Блок дијаграм за целобројни бројач

Решење задатка 8.2.10

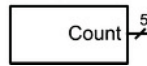
```
Library IEEE;
use IEEE.std_logic_1164.all;
entity counter_integer_up is
    port (Count : integer);
end entity;

architecture counter_integer_up_arch of counter_integer_up is
begin
    counter_integer_up_Proc : process
    signal i : integer;
    begin
        for i in 0 to 31 loop
            Count <= i;
            wait for 10 ns;
        end loop;
    end process;
end architecture;
```

Задатак 8.2.11

Написати VHDL модел за бројач коришћењем for петљу са излазним типом std_logic_vector(4 downto 0). Слика 8.10 приказује блок дијаграм за дефиницију ентитета. Бројач треба да се инкрементира од 00000₂ до 11111₂ а онда да почне поново. Користити wait наредбу у оквиру процеса да би се бројач ажурира на ваких 10ns. Размотрити коришћење променљиве у петљи која генерише вредност бројача, а онда функцију за конверзију из integer у std_logic_vector. Напомена: Овај модел се не може синтетизовати.

counter_5bit_binary_up.vhd



Слика. 8.8 Блок дијаграм за 5-битни бројач

Решење задатка 8.2.10

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity counter_5bit_binary_up is
    port (Count : std_logic_vector(4 downto 0));
end entity;

architecture counter_5bit_binary_up_arch of counter_5bit_binary_up is
begin
    counter_5bit_binary_up_Proc : process
        signal i : integer;
    begin
        for i in 0 to 31 loop
            Count <= std_logic_vector(to_unsigned(i, 5));
            wait for 10 ns;
        end loop;
    end process;
end architecture;
```

Задатак 8.4.2

Написати VHDL тест бенч за верификацију функционалности система на слици 8.4 коришћењем report и assert наредби. Тест бенч треба да унесе улазни код за вектор ABCD ("0000", "0001"). Тест бенч треба да мења улазни образац сваких 10 ns користећи wait наредбу у оквиру процеса стимулације. Користити report и assert наредбе за излазне поруке о статусу сваког теста у симулационом прозору. За сваки улазни вектор креирајте поруку која означава тренутни улазни вектор који се тестира, резултујући излаз вашег DUT-a (DUT – Device Under Testing), и да ли је излаз DUT-a исправан.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Capital "I"

SystemI.vhd

4

ABCD F

Note that the input to the VHDL model is declared as a 4-bit vector.

Слика. 8.4 System I функционалност

Решење задатка 8.4.2

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;

library STD;
use STD.textio.all;

entity SystemI_TB is
end entity;
```

```

architecture SystemI_TB_arch of SystemI_TB is

    component SystemI
        port (ABCD : in std_logic_vector(3 downto 0);
              F: out std_logic);
    end component;

    signal ABCD_TB : std_logic_vector(3 downto 0);
    signal F_TB : std_logic;

begin

    DUT1 : SystemI port map (ABCD => ABCD_TB,
                             F      => F_TB);

    STIMULUS: process
    begin
        ABCD_TB <= "0000"; wait for 10 ns;
        assert (F_TB='1') report "Test je neuspesan za 0000" severity FAILURE;
        assert (F_TB='0') report "Test je uspesan za 0000" severity NOTE;

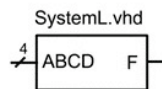
        ABCD_TB <= "0001"; wait for 10 ns;
        assert (F_TB='0') report "Test je neuspesan za 0001" severity FAILURE;
        assert (F_TB='1') report "Test je uspesan za 0001" severity NOTE;
    end process;
end architecture;

```

Задатак 8.5.13

Написати самопроверавајући VHDL тест бенч који учитава тест векторе из екстерне датотеке за верификацију функционалности система на слици 8.7. Креирати улазну датотеку "input_vector.txt" која садржи сваки улазни код за вектор ABCD у редоследу како се појављују у табlici истинитости ("0000", "0001", "0010",...) у посебној линији. Тест бенч треба да учитава сваку линију датотеке посебно и да корисити улазни вектор као улаз DUT-a (DUT – Device Under Testing). Записати излазне резултате у екстерну датотеку "output_vector.txt".

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



Слика. 8.4 System L функционалност

Решење задатка 8.5.13

```

Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;

library STD;
use STD.textio.all;

entity SystemL_TB is
end entity;

architecture SystemL_TB_arch of SystemL_TB is

    component SystemL

```

```

        port (ABCD : in std_logic_vector(3 downto 0);
              F: out std_logic);
    end component;

    signal ABCD_TB : std_logic_vector(3 downto 0);
    signal F_TB : std_logic;

    begin

    DUT1 : SystemL port map (ABCD => ABCD_TB,
                             F      => F_TB);

    STIMULUS: process
        signal i : integer;
        file Fout: TEXT open WRITE_MODE is "output_vector.txt";
        file Fin: TEXT open READ_MODE is "input_vector.txt";
        variable current_wline : line;
        variable current_rline : line;

        begin
            write(currentwline, string'("Input=ABCD, Output=F"));
            writeline(Fout, currentwline);

            for i in 0 to 15 loop
                readline(Fin, current_line);
                read(current_line, ABCD_TB); wait for 10 ns;
                write(currentwline, string'("ABCD="));
                write(currentwline, ABCD_TB);
                write(currentwline, string'(" F="));
                write(currentwline, F_TB);
                writeline(Fout, currentwline)
            end loop;

            end process;
    end architecture;

```