5. Опис секвенцијалних мрежа у VHDL-у

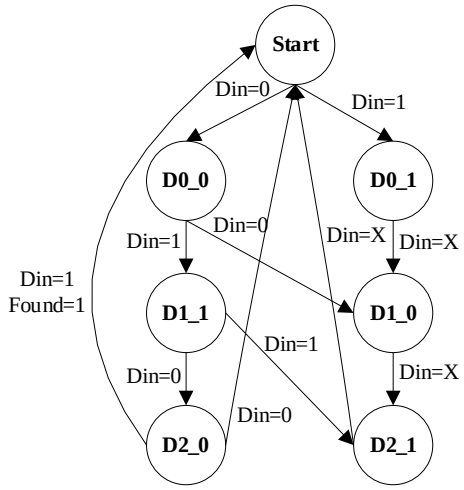
Задатак 9.3.5
Написати VHDL модел понашања за 4-битни серијски бит детектор секвенце сличан оном описаном у примеру 9.11. Користити опис ентитета који је дат на слици 9.3. Користити приступ са 3-процеса за опис коначног аутомата. Улаз за детектор секвенце је означен са Din, а излаз са Found. Детектор ће поставити Found на 1, савки пут када се на улазу појави 4-битна секвенца “0101”. За све друге улазне секвенце излаз има вредност 0. Стања се моделују са корисничко дефинисаним набројивим типом.

Seq_Det_behavioral.vhd

```
entity Seq_Det_behavioral is
  port (Clock, Reset : in  std_logic;
        DIN          : in  std_logic;
        FOUND        : out std_logic);
end entity;
```

Слика. 9.3 Ентитет детектора секвенце

Решење задатка 9.3.5




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

entity Seq_Det_behavioral is
      port (Clock, Reset : in std_logic;
            DIN          : in std_logic;
            FOUND        : out std_logic);
end entity;

architecture Seq_Det_behavioral_arch of Seq_Det_behavioral is
      type State_Type is (Start, D0_0, D0_1, D1_0, D1_1, D2_0, D2_1);
      signal current_state, next_state : State_Type;
begin

      STATE_MEMORY : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        current_state <= Start;
                  elsif (Clock'event and Clock='1') then
                        current_state <= next_state;
                  end if;
      end process;
```

```
    NEXT_STATE_LOGIC : process (current_state, DIN)
          begin
                case (current_state) is
                      when Start => if (DIN = '1') then
                                        next_state <= D0_1;
                                    else
                                        next_state <= D0_0;
                                    end if;
                      when D0_0 =>  if (DIN = '1') then
                                        next_state <= D1_1;
                                    else
                                        next_state <= D1_0;
                                    end if;
                      when D1_1 =>  if (DIN = '1') then
                                        next_state <= D2_1;
                                    else
                                        next_state <= D2_0;
                                    end if;
                      when D0_1 =>  next_state <= D1_0;
                      when D1_0 =>  next_state <= D2_1;
                      when D2_0 =>  next_state <= Start;
                      when D2_1 =>  next_state <= Start;
                      others    =>  next_state <= Start;
                end case;
    end process;

    OUTPUT_LOGIC : process (current_state, DIN)
          begin
                case (current_state) is
                      when D2_0   => if (DIN = '1') then
                                          FOUND <= '1';
                                     else
                                          FOUND <= '0';
                                     end if;
                      when others => FOUND <= '0';
                end case;
    end process;

end architecture;
```

Задатак 9.3.6

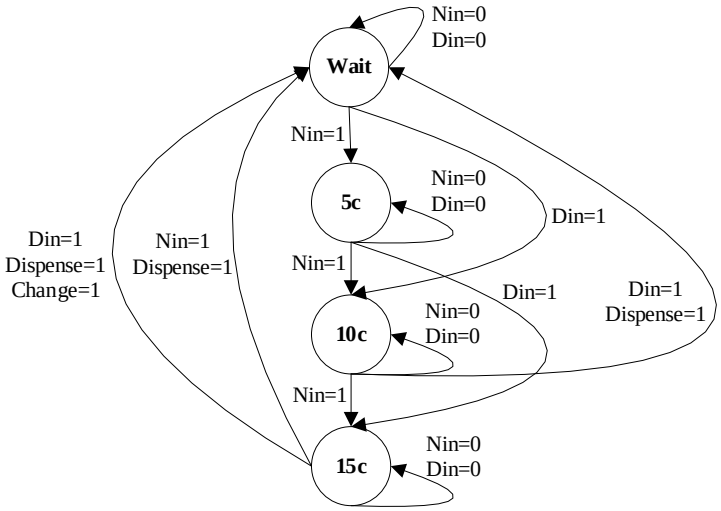
Написати VHDL модел понашања за контролер за 20-центни аутомат за слаткише сличан оном описаном у примеру 9.4. Користити опис ентитета који је дат на слици 9.4. Користити приступ са 3-процеса за опис коначног аутомата. Улаз у контролер су кованице од 5 и 10 центи, а слаткиш се издаје сваки пут када корисник унесе 20 центи. Аутомат има два улаза Nin и Din. Nin се поставља на 1 када корисник убаци кованицу од 5 центи, док се улаз Din поставља на 1 када је убачено 10 центи. Аутомат има 2 излаза: Dispense и Change. Dispense се поставља на 1 сваки пут када корисник убаци укупно најмање 20 центи, док се Change поставља на 1 ако је убачено више од 20 центи и потребно је да се врати кусур од 5 центи. Стања се моделују са корисничко дефинисаним набројивим типом.

Vending_behavioral.vhd

```
entity Vending_behavioral is
  port (Clock, Reset     : in  std_logic;
        Nin, Din         : in  std_logic;
        Dispense, Change : out std_logic);
end entity;
```

Слика. 9.4 Ентитет за аутомат за слаткише

Решење задатка 9.3.6

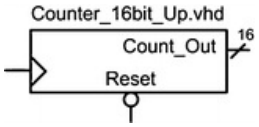



```
Library IEEE;
use IEEE.std_logic_1164.all;

entity Vending_behavioral is
      port (Clock, Reset     : in std_logic;
            Nin, Din         : in std_logic;
            Dispense, Change : out std_logic);
end entity;

architecture Seq_Det_behavioral_arch of Seq_Det_behavioral is
      type State_Type is (sWait, s5c, s10c, s15c);
      signal current_state, next_state : State_Type;
begin

      STATE_MEMORY : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        current_state <= sWait;
                  elsif (Clock'event and Clock='1') then
                        current_state <= next_state;
                  end if;
      end process;

      NEXT_STATE_LOGIC : process (current_state, Nin, Din)
            begin
                  case (current_state) is
                        when sWait => if (Nin = '1' and Din = '0') then
                                          next_state <= s5c;
                                      elsif (Din = '1' and Nin = '0') then
                                          next_state <= s10c;
                                      else
                                          next_state <= sWait;
                                      end if;
                        when s5c   => if (Nin = '1' and Din = '0') then
                                          next_state <= s10c;
                                      elsif (Din = '1' and Nin = '0') then
                                          next_state <= s15c;
                                      else
                                          next_state <= s5c;
                                      end if;
                        when s10c  => if (Nin = '1' and Din = '0') then
                                          next_state <= s15c;
                                      elsif (Din = '1' and Nin = '0') then
                                          next_state <= sWait;
                                      else
```

```
                                    next_state <= s10c;
                                end if;
                        when s15c  => if ((Nin = '1' and Din = '0') or
                                    (Din = '1' and Nin = '0') then
                                    next_state <= sWait;
                                else
                                    next_state <= s15c;
                                end if;
                        others     =>  next_state <= sWait;
                end case;
        end process;

        OUTPUT_LOGIC : process (current_state, Nin, Din)
              begin
                    case (current_state) is
                        when s10c   => if (Din = '1' and Nin = '0') then
                                    Dispense = 1; Change = 0;
                                else
                                    Dispense = 0; Change = 0;
                                end if;
                        when s15c  => if (Din = '1' and Nin = '0') then
                                    Dispense = 1; Change = 0;
                                elsif (Nin = '1' and Din = '0') then
                                    Dispense = 1; Change = 1;
                                else
                                    Dispense = 0; Change = 0;
                                end if;
                        others     =>  Dispense = 0; Change = 0;
                    end case;
        end process;

end architecture;
```

Задатак 9.4.1

Написати VHDL модел понашања за 16-битни бинарни бројач навише коришћењем једног процеса. Блок дијаграм за дефиницију ентитета је приказан на слици 9.6. У моделу декларисати да Count_Out буде типа unsigned и имплементирати интерну функционалност бројача са сигналом типа unsigned.



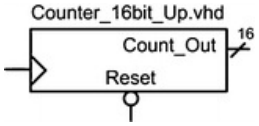
Слика. 9.6 Блок дијаграм за 16-битни бинарни бројач навише

Решење задатка 9.4.1

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

entity Counter_16bit_Up is
      port (Clock, Reset : in std_logic;
            CNT          : out unsigned(15 downto 0);
end entity;

architecture Counter_16bit_Up_arch of Counter_16bit_Up is

      signal CNT_tmp : unsigned(15 downto 0);
      begin

      COUNTER : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        CNT_tmp <= "0000000000000000";
                  elsif (Clock'event and Clock='1') then
```

```
                        CNT_tmp <= CNT_tmp + 1;
                  end if;
      end process;

      CNT <= CNT_tmp;
end architecture;
```

Задатак 9.4.3
Написати VHDL модел понашања за 16-битни бинарни бројач навише коришћењем једног процеса. Блок дијаграм за дефиницију ентитета је приказан на слици 9.6. У моделу декларисати да Count_Out буде типа std_logic_vector и имплементирати интерну функционалност бројача са сигналом типа integer.

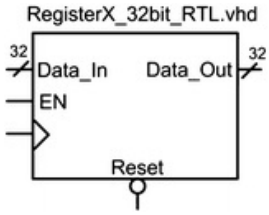


Слика. 9.6 Блок дијаграм за 16-битни бинарни бројач навише

Решење задатка 9.4.3

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.numeric_std_unsigned.all;

entity Counter_16bit_Up is
      port (Clock, Reset : in std_logic;
            CNT          : out std_logic_vector(15 downto 0);
end entity;

architecture Counter_16bit_Up_arch of Counter_16bit_Up is

      signal CNT_int : integer range 0 to 65535;
      begin

      COUNTER : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        CNT_int <= 0;
                  elsif (Clock'event and Clock='1') then
                        if (CNT_int = 65535) then
                              CNT_int <= 0;
                        else
                              CNT_int <= CNT_int + 1;
                        end if;
                  end if;
      end process;

      CNT <= std_logic_vector(to_unsigned(CNT_int,16));
end architecture;
```

Задатак 9.5.3
Написати VHDL модел за 32-битни синхрони регистар. Блок дијаграм за дефиницију ентитета је приказан на слици 9.10. Регистар има улаз за дозволу синхронизације (EN). Регистар треба моделовати коришћењем једног процеса.



Слика. 9.6 Блок дијаграм за 32-битни регистар

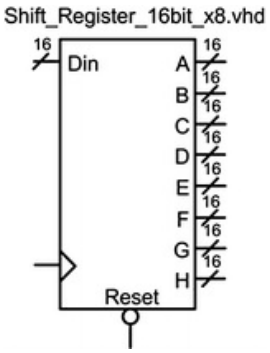
Решење задатка 9.5.3

$\overline{R}$	Clk	EN	Reg_Out	
0	X	X	x"00"	Reset
1	X	0	Last Reg_Out	Disabled (ignore clock)
1	0	1	Last Reg_Out	Store
1	1	1	Last Reg_Out	Store
1	↓	1	Reg_In	Update

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

entity reg is
      port (Clock, Reset, EN : in std_logic;
            Data_In  : in std_logic_vector(31 downto 0);
            Data_Out : out std_logic_vector(31 downto 0);
end entity;

architecture reg_arch of reg is
      begin

      Reg_Proc : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        Data_Out <= x"00";
                  elsif (Clock'event and Clock='1') then
                        if (EN = '1') then
                              Data_Out <= Data_In;
                        end if;
                  end if;
      end process;

end architecture;
```

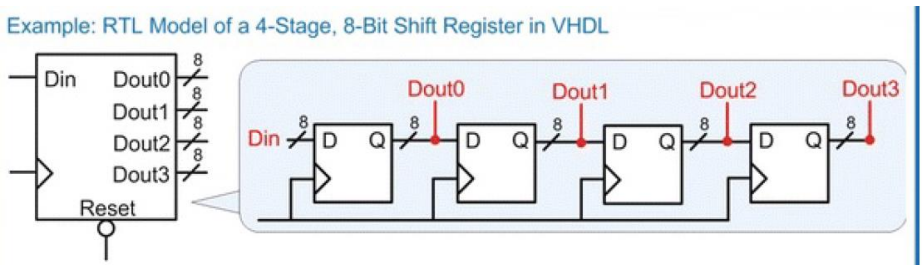
Задатак 9.5.4

Написати VHDL модел за 8-степени 16-битни померачки регистар. Блок дијаграм за дефиницију ентитета је приказан на слици 9.11. Сваки степен померачког регистра има свој излаз (A, B, C, D, E, F, G и H). Користити std_logic и std_logic_vector за све портове.



Слика. 9.6 Блок дијаграм за 16-битни померачки регистар

Решење задатка 9.5.3



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

entity Shift_Register is
      port (Clock, Reset           : in std_logic;
            Din                    : in std_logic_vector(15 downto 0);
            A, B, C, D, E, F, G, H : out std_logic_vector(15 downto 0);
end entity;

architecture Shift_Register_arch of Shift_Register is
      signal DA, DB, DC, DD, DE, DF, DG, DH: std_logic_vector(15 downto 0);
      begin

      Reg_Proc : process (Clock, Reset)
            begin
                  if (Reset='0') then
                        DA <= x"00"; DB <= x"00"; DC <= x"00"; DD <= x"00";
                        DE <= x"00"; DF <= x"00"; DG <= x"00"; DH <= x"00";
                  elsif (Clock'event and Clock='1') then
                        DA <= Din; DB <= DA; DC <= DB; DD <= DC;
                        DE <= DD; DF <= DE; DG <= DF; DH <= DG;
                  end if;
      end process;
      A <= DA; B <= DB; C <= DC; D <= DD; E <= DE; F <= DF; G <= DG; H <= DH;

end architecture;
```