PROIECTAREA SISTEMELOR NUMERICE

TIMER

Proiectat si implementat de catre:

Gog Rares-Flaviu

Grupa 30414

Laborant: Cristian Gabriel Albu

CUPRINS

CONTENTS

SPECIFICATIA PROIECTULUI

Să se proiecteze un timer cu următoarea funcţionalitate: dispozitivul are 4 afişaje BCD - 7 segmente. Primele două afişaje sunt pentru minute, următoarele două pentru secunde. Astfel, valoarea maximă care poate fi afişată este de 99 minute şi 59 secunde.

Dispozitivul are 3 butoane:

*M (de la Minute)

*S (de la Secunde)

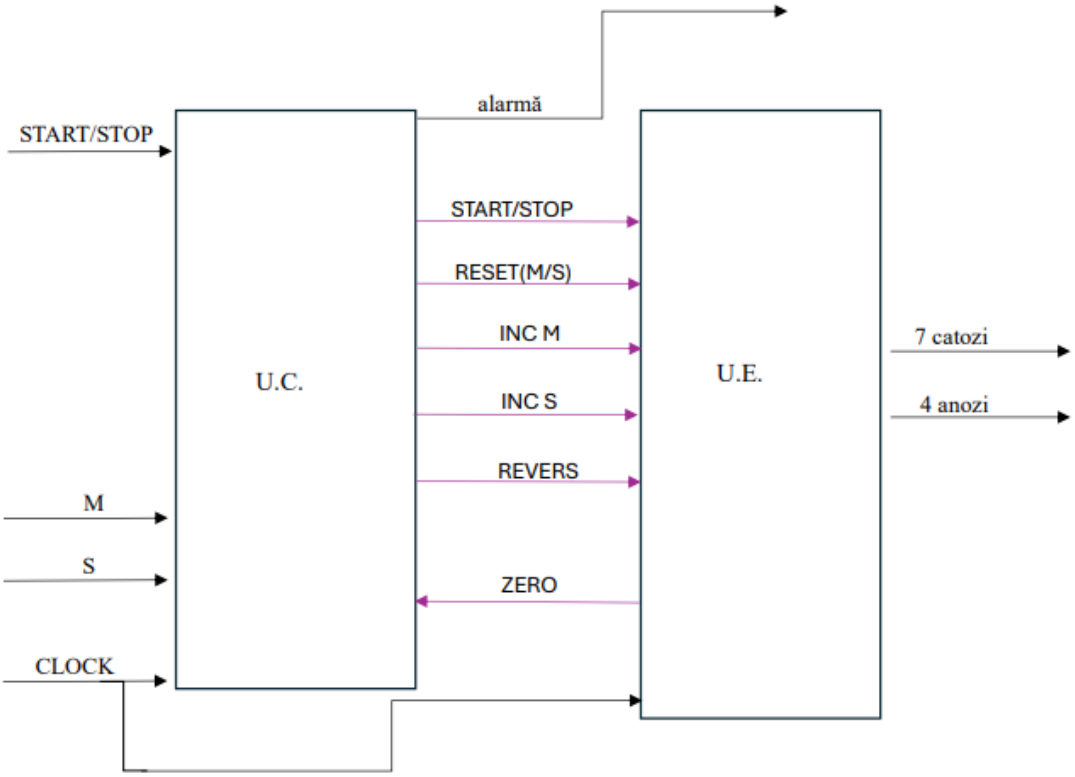
*START / STOP.

Presupunând că iniţial este în starea ZERO, dacă se apasă butonul START / STOP, timerul începe să numere crescător. Dacă se apasă din nou butonul START / STOP, timerul se opreşte la valoarea atinsă în momentul respectiv. Dacă se apasă din nou butonul START / STOP, timerul continuă să numere etc. Dacă ajunge la 99 de minute şi 59 de secunde, urmează din nou ZERO. Dacă se apasă simultan butoanele M (de la Minute) şi S (de la Secunde), timerul se resetează (devine ZERO).

În orice stare, dacă se apasă butonul M, se va incrementa şi afişa valoarea minutelor. În orice stare, dacă se apasă butonul S, se va incrementa şi afişa valoarea secundelor. O dată ce s-a setat o valoare pentru minute şi / sau secunde (prin apăsarea butoanelor M sau S), când se apasă butonul START / STOP, timerul începe să numere descrescător de la valoarea curentă „Minute / Secunde” până la ZERO, iar când se ajunge în starea ZERO se emite un semnal sonor (alarmă) pentru o perioadă de timp care poate fi setată iniţial.

Se consideră că există disponibil un semnal periodic cu frecvenţa de 1 Hz. Proiectul va fi realizat de 1 student.

SCHEMA BLOC


alarmă
START/STOP
START/STOP
RESET(M/S)
INC M
INC S
REVERS
ZERO
U.C.
U.E.
7 catozi
4 anozi
M
S
CLOCK

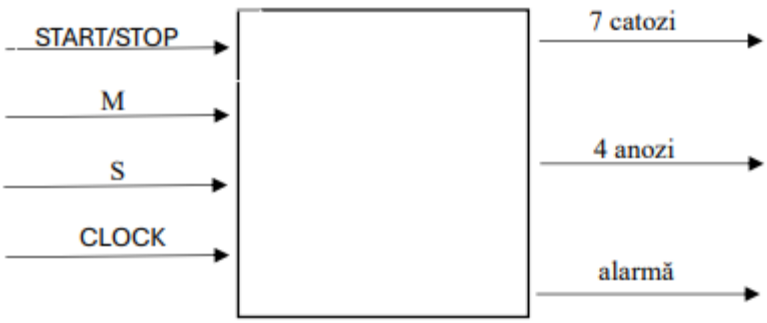
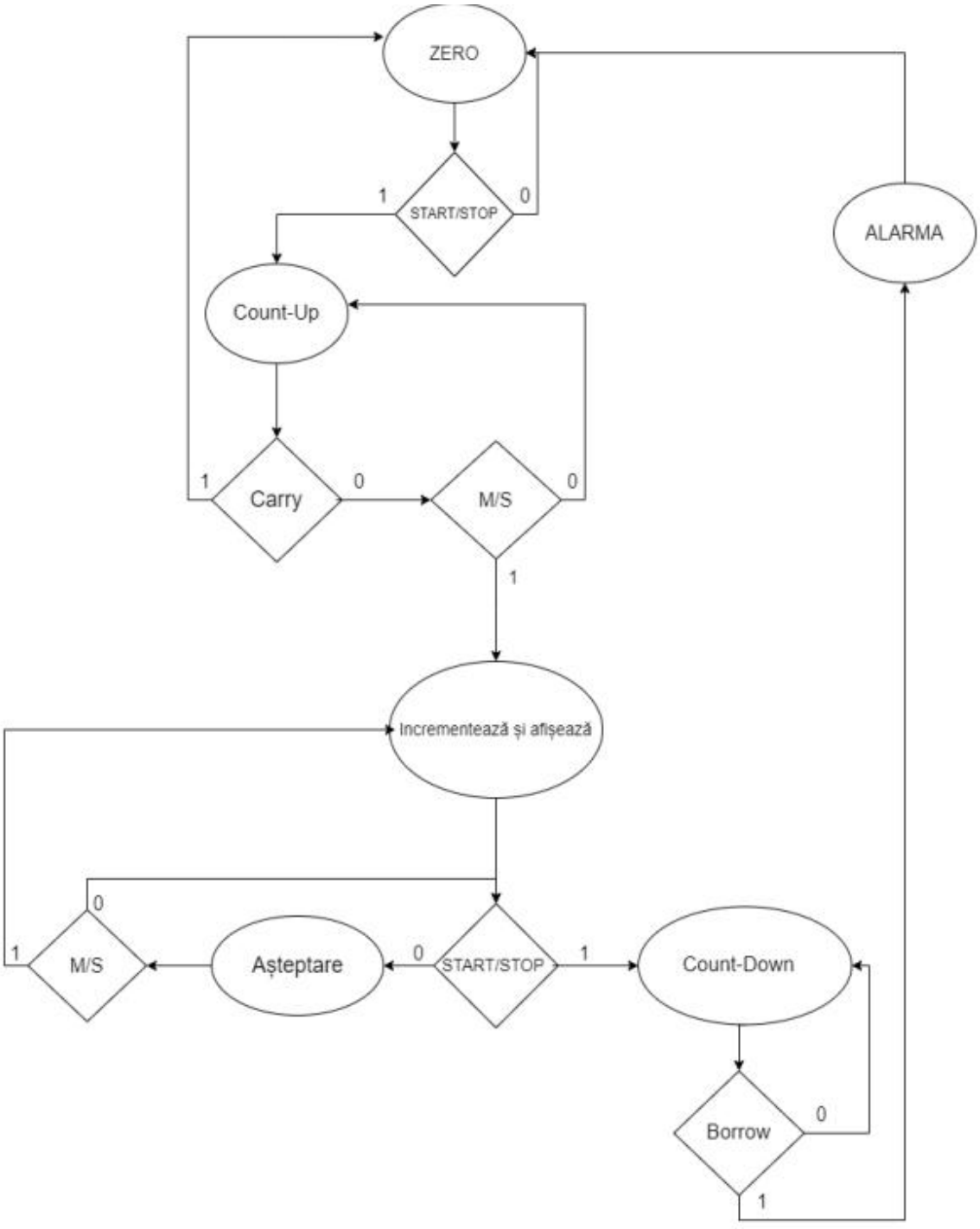

START/STOP
M
S
CLOCK
7 catozi
4 anozi
alarmă

DIAGRAMA PROIECTULUI



COMPONENTE UTILIZATE

1. Numarator modulo 10 (3 componente)
2. Numarator modulo 6 (1 componenta)
3. Debouncer (3 componente)
   *2 debouncere mentin valoarea 1
   *1 debouncer are valoarea 1 o singura data

4. Display : 7-SEGMENT + Anode picker (1 componenta)
5. Clock Divider 100MHZ –> 1HZ (1 componenta)
6. Unitatea de comanda (1 componenta)

CODURILE COMPONENTELOR

NUMARATOR MODULO 10

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.numeric_std.all;
4
5  entity MOD10CTR is
6      Port ( clk : in   STD_LOGIC;
7             btn : in   std_logic;
8             rst: in    std_logic;
9             revers: in std_logic;
10            enable: in std_logic;
11            carry_borrow : out std_logic;
12            counterOUT : out STD_LOGIC_VECTOR (3 downto 0));
13 end MOD10CTR;
14
15 architecture Behavioral of MOD10CTR is
16     signal number : unsigned(3 downto 0);
17 begin
18 process(clk)
19 begin
20     if rising_edge(clk) then
21         if rst = '1' then
22             number <= (others => '0');
23             carry_borrow <= '0';
24
25         elsif enable = '1' then
26             if revers = '1' then
27
28                 if number = "0001" then
29                    number <= number - 1;
30                    carry_borrow <='1';
31                elsif number = "0000" then
```

```
                elsif number = "0000" then
                   number <= "1001";
                   carry_borrow <= '0';
             else
                number <= number - 1;
                carry_borrow <= '0';
             end if;

            elsif revers = '0' then
                if number = "1000" then
                    number <= number + 1;
                    carry_borrow <='1';
                    elsif number = "1001" then
                    number <= (others => '0');
                    carry_borrow <= '0';
                else
                    number <= number + 1;
                    carry_borrow <= '0';
                end if;

           end if;
        elsif btn = '1' then
        if number = "1000" then
                    number <= number + 1;
                    carry_borrow <='1';
                    elsif number = "1001" then
                    number <= (others => '0');
                    carry_borrow <= '0';
                else
                    number <= number + 1;
                    carry_borrow <= '0';
                end if;
           end if;
        end if;
end process;

counterOUT <= std_logic_vector(number);

end Behavioral;
```

In numaratorul modulo 10, denumit „MOD10CTR” , avem ca intrari:

- Clk -> un ceas cu frecventa de 1hz
- Btn -> daca este cazul , butonul pentru incrementarea minutelor/secundelor
- Rst -> semnal utilizat pentru pentru resetare
- Revers -> semnal utilizat pentru determinarea mersului invers
- Enable -> semnal utilizat pentru a functionarea numaratorului

Iesirile din numarator sunt:

- Carry_borrow -> semnal utilizat pentru determinarea folosirii urmatorului numarator
- counterOUT -> semnal cu marimea de 4 biti pentru a fi utilizat la afisarea cifrei

De asemenea este declarat un semnal, „number”, prin intermediul caruia vom numara in sus sau in jos.

Procesul localizat in acest numarator functioneaza pe baza clock-ului de 1HZ. In cazul in care semnalul acestuia este 1 se pot intampla urmatoarele lucruri:

- daca semnaul de resetare este 1 (rst = ’1’) atunci numararea se reseteaza, semnalul de 4 biti si semnalul de carry/borrow fiind setate la 0 ;
- daca semnalul de enable este 1 (enable = ’1’) atunci, pe baza semnaulului „revers” se va numara crescator de la 0 la 9, respectiv descrescator de la 9 la 0; semnalul „carry_borrow” are valoare 1 in cazul in care se ajunge la 8, respectiv 1, pentru a transmite carry/borrow-ul urmatorului numarator (valorile 1 si 8 au fost alese pentru a nu se produce un delay intre numaratoare si ca acestea sa numere corespunzator cerintei) ;
- in cazul in care semnalul btn este activ prin intermediul apasarii butonului de pe placa se va incrementa in sus din punctul in care se afla numaratorul inaintea apasarii butonului (semnaul carry_borrow functioneaza identic cu explicatia de mai sus) ;
- la finalul numaratorului semnalul counterOUT primeste cifra in biti pentru a fi transmisa mai departe urmatoarei componente ;

NUMARATOR MODULO 6

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.numeric_std.all;
4
5  entity MOD6CTR is
6      Port ( clk : in   STD_LOGIC;
7             rst: in    std_logic;
8             revers: in std_logic;
9             enable: in std_logic;
10            carry_borrow: out std_logic;
11            counterOUT : out STD_LOGIC_VECTOR (3 downto 0));
12 end MOD6CTR;
13
14 architecture Behavioral of MOD6CTR is
15     signal number : unsigned(3 downto 0);
16 begin
17
18 process(clk)
19 begin
20     if rising_edge(clk) then
21         if rst = '1' then
22             number <= (others => '0');
23             carry_borrow <= '0';
24         elsif enable = '1' then
25             if revers = '1' then
26                 if number = "0001" then
27                     number <= number - 1;
28                     carry_borrow <='1';
29                elsif number = "0000" then
30                 number <= "0101";
31                 carry_borrow <= '0';
```

```
32              else
33                  number <= number - 1;
34                  carry_borrow <= '0';
35              end if;
36
37              else
38              if number = "0100" then
39                      number <= number + 1;
40                      carry_borrow <='1';
41                  elsif number = "0101" then
42                  number <= (others => '0');
43                  carry_borrow <= '0';
44              else
45                  number <= number + 1;
46                  carry_borrow <= '0';
47              end if;
48              end if;
49          end if;
50          end if;
51  end process;
52
53  counterOUT <= std_logic_vector(number);
54
55  end Behavioral;
56
```

In numaratorul modulo 6, denumit „MOD6CTR” , avem ca intrari:

- Clk -> un ceas cu frecventa de 1hz
- Rst -> semnal utilizat pentru pentru resetare
- Revers -> semnal utilizat pentru determinarea mersului invers
- Enable -> semnal utilizat pentru a functionarea numaratorului

Iesirile din numarator sunt:

- Carry_borrow -> semnal utilizat pentru determinarea folosirii urmatorului numarator
- counterOUT -> semnal cu marimea de 4 biti pentru a fi utilizat la afisarea cifrei

De asemenea este declarat un semnal, „number”, prin intermediul caruia vom numara in sus sau in jos.

Procesul localizat in acest numarator functioneaza pe baza clock-ului de 1HZ. In cazul in care semnalul acestuia este 1 se pot intampla urmatoarele lucruri:

- daca semnaul de resetare este 1 (rst = ’1’) atunci numararea se reseteaza, semnalul de 4 biti si semnalul de carry/borrow fiind setate la 0 ;
- daca semnalul de enable este 1 (enable = ’1’) atunci, pe baza semnaulului „revers” se va numara crescator de la 0 la 5, respectiv descrescator de la 5 la 0; semnalul „carry_borrow” are valoare 1 in cazul in care se ajunge la 4, respectiv 1, pentru a transmite carry/borrow-ul urmatorului numarator (valorile 1 si 4 au fost alese pentru a

nu se produce un delay intre numaratoare si ca acestea sa numere corespunzator cerintei) ;

- la finalul numaratorului semnalul counterOUT primeste cifra in biti pentru a fi transmisa mai departe urmatoarei componente ;

DIVIZOR DE FRECVENTA

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Clock_divider is
5      Port ( clk_intrare : in STD_LOGIC;
6             clk_1hz : out STD_LOGIC);
7  end Clock_divider;
8
9  architecture Behavioral of clock_divider is
10 signal count_up : integer :=0;
11 signal  b : std_logic :='0';
12 begin
13 process(b,clk_intrare)
14 begin
15 if(rising_edge(clk_intrare)) then
16     count_up <= count_up + 1;
17         if(count_up = 49999999) then
18             b <= not b;
19             count_up <= 0;
20         end if;
21 end if;
22
23 clk_1hz <= b;
24
25 end process;
26 end Behavioral;
```

In acest divizor de frecventa, denumit „clock_divider”, avem ca intrarea clock-ul placutei de 100MHZ si ca iesire, un nou clock de 1HZ. Avem 2 semnale declarate in arhitectura componentei:

- count_up, de tip integer, prin intermediul acesteia vom numara in sus pentru a diviza frecventa clock-ului
- b, de tip std_logic, prin care vom transmite noului clock semnalul corespunzator pentru a functiona corect

Procesul localizat in aceasta componenta functioneaza pe baza clock-ului de 100MHZ. In cazul in care avem „rising edge” la acest clock se va numara in sus pana se ajunge la ~50 milioane. Cand se ajunge la aceasta valoare semnalul „b” va primi opusul lui si numararea se va reseta. La finalul componentei noul clock, „clk_1hz” primeste valoarea lui b.

DISPLAY

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity SVN_SEG09 is
    Port (  clk : in STD_LOGIC;
            Rez : in STD_LOGIC_VECTOR (15 downto 0)
            an : out STD_LOGIC_VECTOR (3 downto 0);
            cat : out STD_LOGIC_VECTOR (6 downto 0)
end SVN_SEG09;

architecture Behavioral of SVN_SEG09 is
signal counter: std_logic_vector (15 downto 0);
signal n: std_logic_vector (3 downto 0);

begin
process(CLK)
begin
    if clk'event and clk='1' then
        counter <= counter + 1;
    end if;
end process;

process(n)
begin

    case (n) is
        when "0001" => cat <= "1111001"; --1
        when "0010" => cat <= "0100100"; --2
        when "0011" => cat <= "0110000"; --3
        when "0100" => cat <= "0011001"; --4
        when "0101" => cat <= "0010010"; --5
        when "0110" => cat <= "0000010"; --6
        when "0111" => cat <= "1111000"; --7
        when "1000" => cat <= "0000000"; --8
        when "1001" => cat <= "0010000"; --9
        when others => cat <= "1000000"; --0
    end case;

end process;

process(counter(15 downto 14))
    begin
        case counter(15 downto 14) is
         when "00" => an <= "1110";
         when "01" => an <= "1101";
         when "10" => an <= "1011";
         when others => an <= "0111";
      end case;
end process;
```

```
51
52  process(counter(15 downto 14),Rez)
53      begin
54          case counter (15 downto 14) is
55              when "00"=> n <= Rez(3 downto 0);
56              when "01"=> n <= Rez(7 downto 4);
57              when "10"=> n <= Rez(11 downto 8);
58              when others => n <= Rez(15 downto 12);
59          end case;
60  end process;
61
62
63
64  end Behavioral;
65
```

In aceasta componenta de afisare pe placa, denumita „SVN_SEG09”, avem ca intrari:

- Clk -> clock-ul intern al placii
- Rez -> semnal de 16 biti prin intermediul caruia sunt primite cele 4 cifre in BCD

Iesirile din aceasta componenta sunt:

- An -> semnal de 4 biti prin intermediul caruia se alege anodul
- Cat -> semnal de 7 biti prin intermdeiul caruia se alege catodul

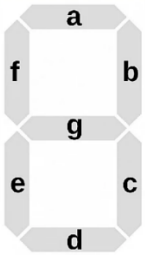
De asemenea, sunt declarate 2 semnale:

- Counter -> semnal utilizat pentru alegerea anodului si a cifrei
- n -> semnal utilizat pentru alegerea cifrei corespunzatoare

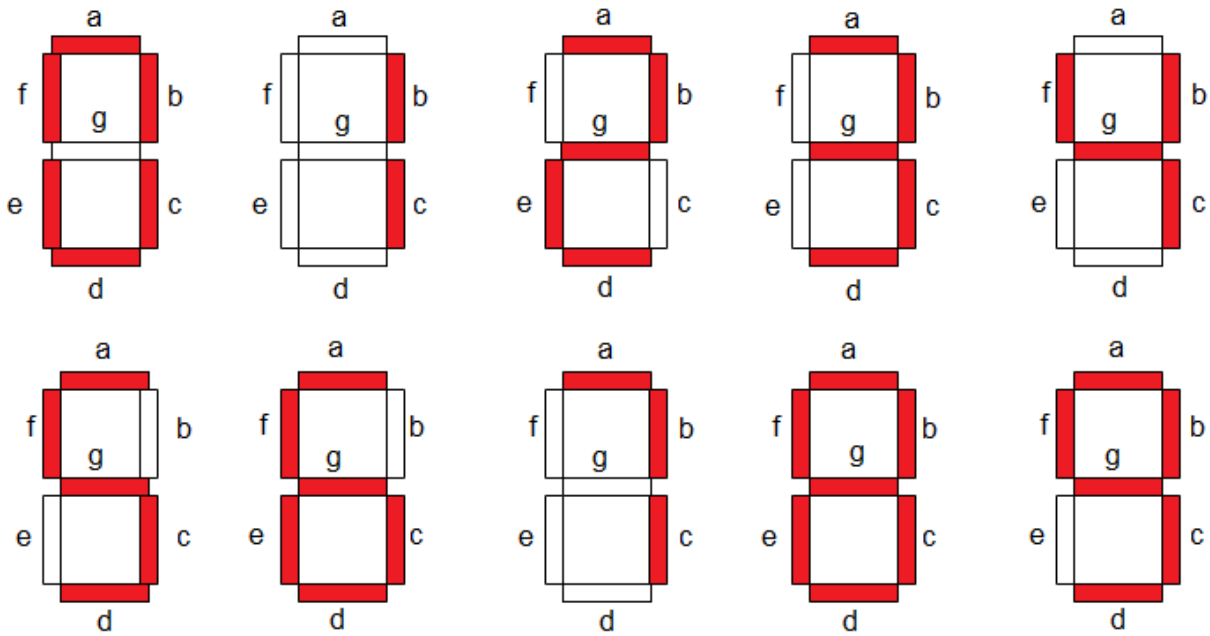
In aceasta componeta avem 4 procese:

Primul proces numara in sus prin intermediul semnalului counter, pe baza clock-ului primit ca intrare in componenta.

Al doilea proces realizeaza cifra corespunzatoare fiecarui caz primit in semnalul „n”. De exemplu, daca „n” este 0101 (5) atunci semnalul primit de cat, in logica negativa, va fi 0010010 (gfedcba). In urmatoarea imagine este reprezentat fiecare catod cu litera (functia) corespunzatoare.



Astfel, va fi reprezentata fiecare cifra in functie de semnalul primit in „n”.

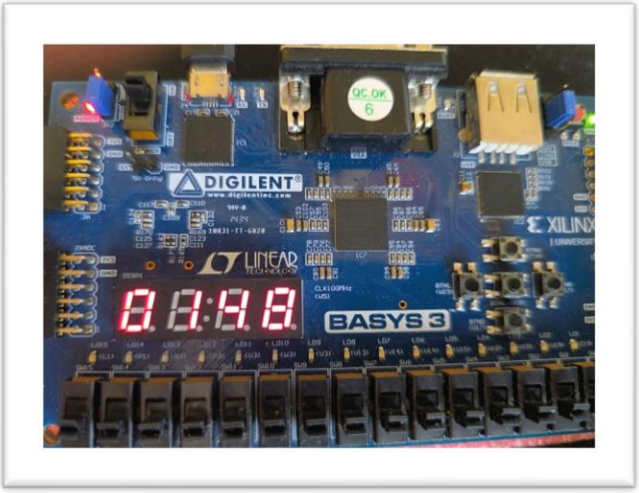


In al treilea proces, prin intermediul ultimilor 2 biti ai semnalului „counter”, „an” primeste valoarea anodului corespunzator acestor biti.

In al patrulea proces, tot prin intermediul ultimilor 2 biti ai semnalului „counter”, „n” va primi cifra corespunzatoare in BCD.

Astfel, prin intermediul ultimelor doua procese, ficare anod va primi cifra care ii conrespunde. De exemplu, pentru primul anod „an” este 1110 si „n” va lua valoarea primilor 4 biti ai semnalului „Rez”.

In final, prin intermediul acestor procese va fi afisat pe display timpul realizat de numaratoare.



DEBOUNCER

1.DebouncerSS

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

entity DebouncerSS is
    Port ( BTN : in STD_LOGIC;
           CLK : in STD_LOGIC;
           ENABLE : out STD_LOGIC);
end DebouncerSS;

architecture Behavioral of debouncerSS is
    signal COUNT: STD_LOGIC_VECTOR(15 downto 0):=(others =>'0');
    signal Q1, Q2, Q3: STD_LOGIC:='0';
begin
   process(CLK)
        begin
            if (CLK='1' and CLK'EVENT) then
                COUNT<=COUNT+1;
            end if;
    end process;

   process(CLK)
        begin
            if (CLK='1' and CLK'EVENT) then
                if COUNT=x"FFFF" then
                    Q1<=BTN;

                end if;
            end if;
    end process;

   process(CLK)
        begin
            if (CLK='1' and CLK'EVENT) then
                    Q2<=Q1;
                    Q3<=Q2;
            end if;
    end process;

   enable <= Q2 and not Q3;

end Behavioral;
```

In aceasta componenta, denumita „DebouncerSS”, avem urmatoarele intrari:

- btn -> semnaul transmis de buton in momentul apasarii
- Clk -> clock-ul intern al placii

Ca iesire avem semnaulul „Enable” care va fi butonul trecut prin acest Debouncer. De asemenea, avem semnale declarate precum : count, Q1, Q2 si Q3.

Prin intermediul acestui proces am realizat 3 bistabile. Primele 2 vor fi 1 cat timp este apasat butonul, iar ultimul va fi 1 cand se va ridica degetul de pe buton pentru o perioada scurta de timp. Astfel, cand se va apasa butonul enable va lua valoare 1 o singura data si in acest mod putem considera ca butonul a fost apasat o singura data.

Acest tip de debouncer va fi folosit numai pentru butonul Start/Stop.

2.DebouncerSM

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  entity DebouncerSM is
6      Port ( BTN : in STD_LOGIC;
7             CLK : in STD_LOGIC;
8             ENABLE : out STD_LOGIC);
9  end DebouncerSM;
10
11 architecture Behavioral of DebouncerSM is
12     signal COUNT: STD_LOGIC_VECTOR(15 downto 0):=(others =>'0');
13     signal Q1, Q2, Q3: STD_LOGIC:='0';
14     signal EN1, EN2: STD_LOGIC;
15 begin
16    process(CLK)
17         begin
18             if (CLK='1' and CLK'EVENT) then
19                 COUNT<=COUNT+1;
20             end if;
21     end process;
22
23    process(CLK)
24         begin
25             if (CLK='1' and CLK'EVENT) then
26                 if COUNT=x"FFFF" then
```

```
27                      Q1<=BTN;
28                  end if;
29              end if;
30      end process;
31
32      process(CLK)
33          begin
34              if (CLK='1' and CLK'EVENT) then
35                      Q2<=Q1;
36                      Q3<=Q2;
37              end if;
38      end process;
39
40      EN1 <= Q2 and not Q3;
41      EN2 <= not Q1 and not Q2 and Q3;
42      Enable <= EN1 or EN2;
43
44  end Behavioral;
```

In aceasta componenta, denumita „DebouncerSM”, avem urmatoarele intrari:

- btn -> semnaul transmis de buton in momentul apasarii
- Clk -> clock-ul intern al placii

Ca iesire avem semnaulul „Enable” care va fi butonul trecut prin acest Debouncer. De asemenea, avem semnale declarate precum : count, Q1, Q2,Q3,EN1 si EN2.

Prin intermediul acestui proces am realizat 3 bistabile. Primele 2 vor fi 1 cat timp este apasat butonul, iar ultimul va fi 1 cand se va ridica degetul de pe buton pentru o perioada scurta de timp. Prin intermediul semnalelor EN1 si EN2 vom verifica starea bistabilelor. Astfel, butonul va f i considerat 1 cat timp este apasat, valoarea lui fiind 0 in momentul in care ridicam degetul de pe buton. Semnalul „Enable” primind doar o valoare dintre cele doua semnale.

Acest tip de debouncer va fi folosit numai pentru butonoanele de incrementare a minutelor si a secundelor (S si M).

UNITATEA DE CONTROL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Timer is
5    Port (    clk : in std_logic;
6              M : in std_logic;
7              S : in std_logic;
8              SS : in std_logic;
9              bec : out std_logic;
10             an : out std_logic_vector (3 downto 0);
11             ca : out std_logic_vector (6 downto 0));
12 end Timer;
13
14 architecture Behavioral  of Timer is
15
16 component MOD10CTR is
17     Port ( clk : in STD_LOGIC;
18            BTN : IN STD_LOGIC;
19            rst : in std_logic;
20            revers : in std_logic;
21            enable : in std_logic;
22            carry_borrow : out std_logic;
23            counterOUT : out STD_LOGIC_VECTOR (3 downto 0));
24 end component ;
25
26 component  MOD6CTR is
27     Port ( clk : in STD_LOGIC;
28            rst : in std_logic;
29            revers : in std_logic;
30            enable : in std_logic;
31            carry_borrow : out std_logic;
32            counterOUT : out STD_LOGIC_VECTOR (3 downto 0));
33 end component ;
34
35 component DebouncerSM is
36     Port ( BTN : in STD_LOGIC;
37            CLK : in STD_LOGIC;
38            ENABLE : out STD_LOGIC);
39 end component ;
40
41 COMPONENT DebouncerSS is
42     Port ( BTN : in STD_LOGIC;
43            CLK : in STD_LOGIC;
44            ENABLE : out STD_LOGIC);
45 end component;
46
47 component SVN_SEG09 is
48     Port (   clk : in std_logic;
49              Rez : in STD_LOGIC_VECTOR (15 downto 0);
50              an : out STD_LOGIC_VECTOR (3 downto 0);
```

```
              cat : out STD_LOGIC_VECTOR (6 downto 0));
end component ;

component   clock_divider is
    Port (  clk_intrare : in STD_LOGIC;
            clk_1hz : out STD_LOGIC);
end component;

signal resetare, invers : std_logic :='0';
signal CB_S0, CB_S1, CB_M0, CB_M1: std_logic;
signal S0EN, S1EN, M0EN, M1EN: std_logic;
signal clk1hz : std_logic;
signal concat_cif : std_logic_vector( 15 downto 0);
signal M_DB, S_DB, SS_DB : STD_LOGIC;
signal btn_state_S, btn_state_SS, btn_state_M : std_logic := '0';
SIGNAL EN1,EN2,EN3,EN4,EN5,EN6,EN7 : STD_LOGIC :='0';
signal prev_state_s, prev_State_m : std_logic :='0';
signal inc_s,inc_m : std_logic;

begin

process (btn_state_s, btn_state_ss, btn_state_m, ss_Db, m_db, s_db, clk)
begin


if rising_edge (clk) then

        if s_db = '1' and resetare = '0' then
            prev_State_s <= '1';
            btn_State_s <= not btn_state_s;
           if(EN7 ='0') THEN btn_state_ss <= '0';
           END IF;

        elsif m_db = '1' and resetare= '0' then
            prev_State_m <= '1';
            btn_state_m <= not btn_state_m;
            if(EN7 ='0') THEN btn_state_ss <= '0';
            END IF;

        elsif ss_db = '1' then
            btn_state_ss <= not btn_state_ss;
            btn_state_s  <= '0';
            btn_state_m  <= '0';
            resetare <= '0';
            if prev_state_m = '1' or prev_state_s = '1' then
                        invers <= '1';
                        EN7 <='1';
                end if;
        end if;


     if s = '1' and m = '1' then
         resetare <= '1';
         invers   <= '0';
         btn_state_s  <= '0';
         btn_state_m  <= '0';
         btn_state_ss <= '0';
         PREV_STATE_S <= '0';
         PREV_STATE_M <= '0' ;
         bec <='0';
         EN7 <='0';
         EN6 <='0';
         EN5 <='0';
         EN4 <='0';
         EN3 <='0';
         EN2 <='0';
         EN1 <='0';
     end if;

     if invers = '1' and concat_cif = "0000000000000000" then
         bec <= '1';
         btn_state_s  <= '0';
         btn_state_m  <= '0';
         btn_state_ss <= '0';
```

```
123         invers <= '0';
124     END IF;
125
126     IF CB_M1 = '1' and CB_M0 = '1' AND CB_S1 = '1' AND CB_S0 = '1' THEN
127         resetare <= '1';
128         invers   <= '0';
129         btn_state_s  <= '0';
130         btn_state_m  <= '0';
131         btn_state_ss <= '0';
132         PREV_STATE_S <= '0';
133         PREV_STATE_M <= '0' ;
134         bec <='0';
135         EN7 <='0';
136         EN6 <='0';
137         EN5 <='0';
138         EN4 <='0';
139         EN3 <='0';
140         EN2 <='0';
141         EN1 <='0';
142     end if;
143
144     if(invers='1' and concat_cif(3 downto 0)="0000" and CB_S0='0' and concat_cif(15 downto 4) /= "000000000000") THEN
145             EN1 <= '1';
146    else EN1<='0';

147    END IF;
148    if(invers='1' and concat_cif(7 downto 4)="0000" and CB_S1='0' and concat_cif(15 downto 8) /="00000000" ) THEN
149             EN2 <= '1';
150    else EN2<='0';
151    END IF;
152    if(invers='1' and concat_cif(11 downto 8)="0000" and CB_M0='0' and concat_cif(15 downto 12) /= "0000") THEN
153             EN3 <= '1';
154    else EN3<='0';
155    END IF;
156
157    if(invers='1' and concat_cif(3 downto 0)="1001" and CB_S0='1') THEN
158             EN4 <= '0';
159    else EN4<='1';
160    END IF;
161    if(invers='1' and concat_cif(7 downto 4)="0101" AND CB_S1='1' ) THEN
162             EN5 <= '0';
163    else EN5<='1';
164    END IF;
165    if(invers='1' and concat_cif(11 downto 8)="1001" and CB_M0='1') THEN
166             EN6 <= '0';
167    else EN6<='1';
168    END IF;
169 end if;
170 end process;

171
172 --enable pentru fiecare counter--
173 S0EN <= btn_state_ss;
174 S1EN <= (S0EN and ((CB_S0 and EN4) OR EN1)) or (((CB_S0 and EN4) OR EN1) and (btn_state_s and NOT EN7) );
175 M0EN <= (S1EN and ((CB_S1 and EN5)  OR EN2));
176 M1EN <= (M0EN and ((CB_M0 and EN6)  OR EN3)) or (((CB_M0 and EN6)  OR EN3) and (btn_state_m and NOT EN7) );
177 INC_S <= btn_state_S and not EN7;
178 INC_M <= btn_state_m and not EN7;
179
180
181 DB1 : DebouncerSS port map (btn => SS, clk => clk, enable => SS_DB );
182 DB2 : DebouncerSM   port map (btn => S , clk => clk, enable =>  S_DB );
183 DB3 : DebouncerSM   port map (btn => M , clk => clk, enable =>  M_DB );
184
185 C1 : clock_divider port map ( clk_intrare => clk, clk_1hz => clk1hz);
186 C2 : MOD10CTR  port map ( clk => clk1hz, btn => inc_S,       rst => resetare, revers => invers, enable => S0EN , carry_borrow => CB_S0, counterOUT => concat_cif( 3 downto  0));
187 C3 : MOD6CTR   port map ( clk => clk1hz,                     rst => resetare, revers => invers, enable => S1EN , carry_borrow => CB_S1, counterOUT => concat_cif( 7 downto  4));
188 C4 : MOD10CTR  port map ( clk => clk1hz, btn => inc_m      , rst => resetare, revers => invers, enable => M0EN , carry_borrow => CB_M0, counterOUT => concat_cif(11 downto  8));
189 C5 : MOD10CTR  port map ( clk => clk1hz, btn => '0'        , rst => resetare, revers => invers, enable => M1EN , carry_borrow => CB_M1, counterOUT => concat_cif(15 downto 12));
190 C6 : SVN_SEG09 port map ( clk => clk, rez => concat_cif, an => an , cat => ca );
191
192 end Behavioral;
193
```

In unitatea de control, denumita „Timer”, avem ca intrari:

- Clk -> clock-ul intern al placii
- S -> butonul pentru incrementarea secundelor si resetare
- M -> butonul pentru incrementarea minutelor si resetare
- Ss -> butonul pentru Start/Stop

Unitatea de control are urmatoarele iesiri:

- An -> semnalul de selectare a anodului
- Cat -> semnalul de selectare a catodului
- Bec -> semnalul de aprindere a becului

In unitatea de control se regasesc toate componentele declarate la inceputul arhitecturii pentru a fi utilizate. De asemenea, avem semnale declarate in aceasta componenta:

- Resetare -> semnal cu nume sugestiv, in cazul in care acest semnal este 1, programul se va reseta
- Invers -> semnal cu nume sugestiv, in cazul in care acest semnal este 1, numărarea se va face in sens invers
- CB_S0 , CB_S1, CB_M0, CB_M1 -> aceste semnale reprezinta carry/borrow -ul fiecarui numarator, daca oricare este 1, acesta va intra in logica de pornire a urmatorului numarator
- S0EN, S1EN, M0EN, M1EN -> semnale care permit numaratoarelor sa functioneze
- Clk1hz -> semnal pentru clock -ul de 1HZ
- Concat_cif -> semnal cu dimensiunea de 16 biti, care permite memorarea cifrelor in BCD pentru a fi transmise componentei de display
- M_DB, S_DB, SS_DB -> semnale ce reprezinta fiecare buton dupa ce a trecut printr-un debouncer
- Btn_state_s , Btn_state_m, Btn_state_SS -> semnale ce reprezinta starea in care se afla butonul desemnat acesteia
- EN1, EN2, EN3, EN4, EN5, EN6, EN7 -> semnale ce primesc valoarea 1 in diferite cazuri pentru a repara anumite erori

In procesul localizat in unitatea de control se vor prelucra mai multe elemente. Aceste elemente vor fi explicate mai jos in functie de liniile de cod unde se regasesc:

- 77 – 98 -> In cazul in care semnalul s_Db este 1 si resetare este 0 atunci programul va memora faptul ca butonul a fost apasat, in prev_state_s, si va schimba starea semnalului btn_state_s pentru a incrementa secundele. De asemenea, daca numararea din start/stop era activa si mergea crescator, se va pune pauza.
  -> In cazul in care semnalul m_Db este 1 si resetare este 0 atunci programul va memora faptul ca butonul a fost apasat, in prev_state_m, si va schimba starea semnalului btn_state_m pentru a incrementa minutele. De asemenea, daca numararea din start/stop era activa si mergea crescator, se va pune pauza.

-> In cazul in care semnalul ss_Db este 1 atunci btn_state_ss isi va schimba schima valoare cu opusul ei. Astfel, numaratoarea porneste sau se opreste. Totodata, incrementarile si resetarea primesc valoarea 0 si, in cazul in care a fost realizata o incrementare, va incepe numaratoarea inversa si EN7 primeste valoarea 1.

- 100 – 116 -> In cazul in care sunt apasate ambele butoane programul se va reseta, toate semnalele primind valoarea 0 si resetare primind valoarea 1.
- 118 – 124 -> In cazul in care numaratoarea inversa ajunge la 00:00 atunci se va aprinde un bec si programul se va opri.
- 126 – 142 -> In cazul in care numaratoarea crescatoare ajunge la 99:59 codul va trece in starea initiala.
- 144 – 155 -> In aceasta parte a codului se verifica, in momentul cand se numara invers si se ajunge la 0, daca exista cifre diferite de 0 in stanga celorlalte cifre si nu exista borrow. Daca se intampla acest lucru atunci EN1, EN2 sau EN3, in functie de pozitia cifrei care a intampinat aceasta problema, va primi valoarea 1.
- 157 – 169 -> In cazul in care se incrementeaza pana la 9, respectiv 5 in cazul secundelor, va fi preluat un carry. Acest carry ramane si daca incepe numaratoarea inversa. Aceasta parte din cod previne acest lucru.

Liniile 172 – 178 prelucreaza datele primite pentru a permite functionarea numaratoarelor. Aceasta parte nu este regasita in proces. De asemenea, in aceast cod se permite incrementarea numai in cazul in care nu functioneaza numaratoarea inversa.

In liniile 181 - 190 sunt utilizate componetele specificate mai sus. Fiecare semnal de intrare/iesire fiind conectat cu un alt semnal astfel incat este realizata conexiunea corecta intrea componente.

INSTRUCTIUNI DE UTILIZARE

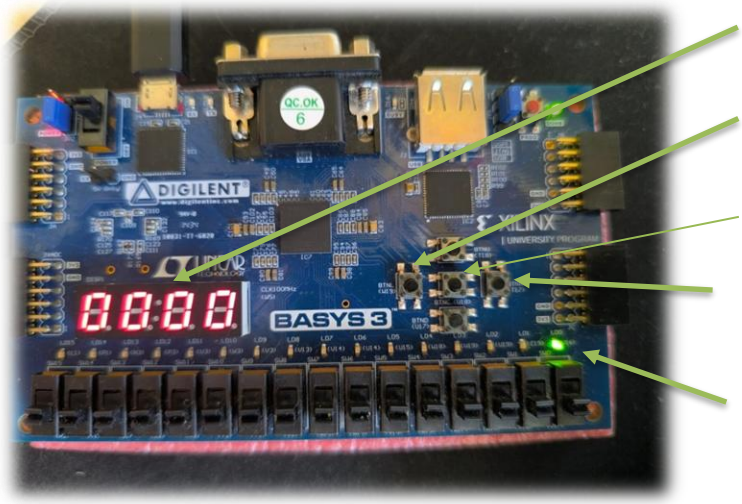
Placa utilizata are 5 butoane, din care proiectul utilizeaza numai 3. Butonul din mijloc este butonul de START/STOP, butonul din stanga este butonul de incrementare a minutelor (M), iar butonul din dreapta este butonul de incrementare a secundelor (S).

Butoanele S si M vor incrementa cate o secunda/minut, la un interval de 1 secunda, numai daca este mentinut degetul apasat pe butonul respectiv. In momentul in care utilizatorul apasa ambele butoane numararea se va reseta. In cazul in care se doreste incrementarea dupa resetare este necesara o apasare pe butonul START/STOP pentru a activa butoanele S si M.

Butonul START/STOP, cat timp nu a fost apast butonul S sau M, in momentul apasarii va incepe numaratoarea crescatoare sau va pune pauza acestei numaratori. In cazul in care a fost apasat unul dintre celelalte doua butoane atunci, in momentul apasarii butonului START/STOP va incepe numaratoarea inversa, butoanele S si M devenind indisponibile incrementarii. In momentul in care se ajunge la 00:00 numaratoarea se va opri si se va aprinde un bec.

Timpul este afisat pe partea digitala a placii timpul fiind reprezentat astfel: (XY:ZT)

Zeci de minute(X) ~ Minute(Y) : Zeci de scunde(Z) ~ Secunde(T)



Afisarea timpului pe display.

Buton pentru incrementarea minutelor.

Butonul START/STOP.

Butonul pentru incrementarea secundelor.

Becul ce se aprinde cand timpul ajunge la 00: 00.

FISIERUL DE CONSTRANGERI

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {bec}]
        set_property IOSTANDARD LVCMOS33 [get_ports {bec}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
```

#set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]

#set_property PACKAGE_PIN U14 [get_ports {led[6]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]

#set_property PACKAGE_PIN V14 [get_ports {led[7]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]

#set_property PACKAGE_PIN V13 [get_ports {led[8]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]

#set_property PACKAGE_PIN V3 [get_ports {led[9]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]

#set_property PACKAGE_PIN W3 [get_ports {led[10]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]

#set_property PACKAGE_PIN U3 [get_ports {led[11]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]

#set_property PACKAGE_PIN P3 [get_ports {led[12]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]

#set_property PACKAGE_PIN N3 [get_ports {led[13]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]

#set_property PACKAGE_PIN P1 [get_ports {led[14]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]

#set_property PACKAGE_PIN L1 [get_ports {led[15]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]

##7 segment display

set_property PACKAGE_PIN W7 [get_ports {ca[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[0]}]

set_property PACKAGE_PIN W6 [get_ports {ca[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[1]}]

set_property PACKAGE_PIN U8 [get_ports {ca[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[2]}]

set_property PACKAGE_PIN V8 [get_ports {ca[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[3]}]

set_property PACKAGE_PIN U5 [get_ports {ca[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[4]}]

set_property PACKAGE_PIN V5 [get_ports {ca[5]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[5]}]

set_property PACKAGE_PIN U7 [get_ports {ca[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {ca[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]

#set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]

set_property PACKAGE_PIN U4 [get_ports {an[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]

set_property PACKAGE_PIN V4 [get_ports {an[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]

set_property PACKAGE_PIN W4 [get_ports {an[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]

##Buttons

set_property PACKAGE_PIN U18 [get_ports SS]

set_property IOSTANDARD LVCMOS33 [get_ports SS]

#set_property PACKAGE_PIN T18 [get_ports btnU]

#set_property IOSTANDARD LVCMOS33 [get_ports btnU]

set_property PACKAGE_PIN W19 [get_ports M]

set_property IOSTANDARD LVCMOS33 [get_ports M]

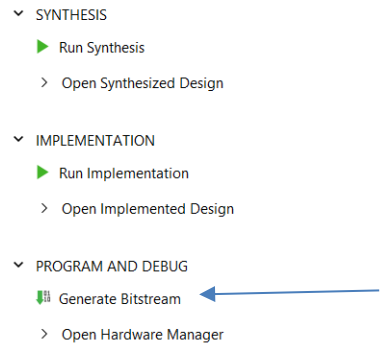
set_property PACKAGE_PIN T17 [get_ports S]

set_property IOSTANDARD LVCMOS33 [get_ports S]

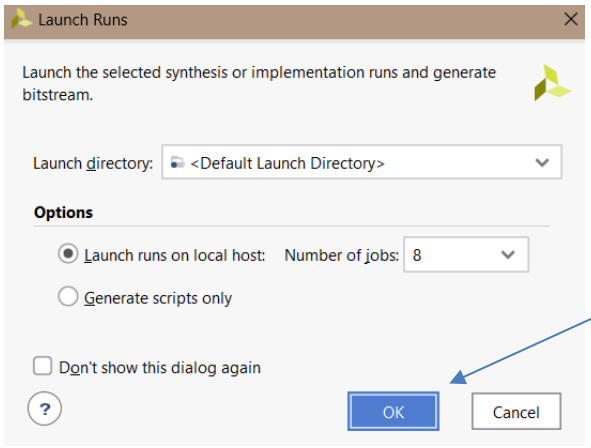
#set_property PACKAGE_PIN U17 [get_ports btnD]

#set_property IOSTANDARD LVCMOS33 [get_ports btnD]

CREAREA FISIERULUI .BIT



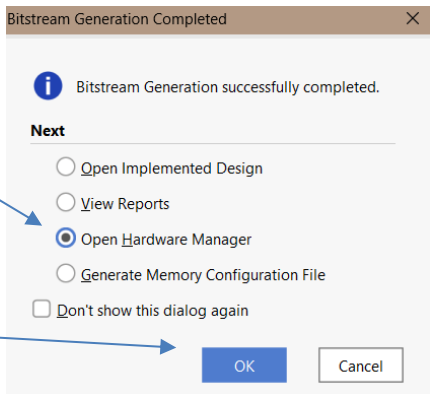
Se apasa butonul "Generate Bitstream"



Apasam butonul "OK"

Asteptam pana se termina prelucrarea datelor introduse.

Selectam "Open Hardware Manager" pentru implementarea pe placa.



Apasam "OK".

Bitstreamul a fost creat cu success.

JUSTIFICAREA SOLUTIEI ALESE

Am ales aceasta solutie deoarece, prin intermediul informatiei primite, aceasta varianta de rezolvare a fost prima optiune l-a care m-am gandit. Pe parcursul proiectului am intampinat dificultati, dar am reusit sa gasesc solutiile potrivite pentru a ajunge la scopul final. Probabil nu este cea mai optima rezolvare a proiectului, dar este functionala.

In concluzie, solutia realizata este functionala, dar poate fi modificata intr-o varianta simplificata.

POSIBILITATI DE DEZVOLTARE ULTERIOARE

Din punct de vedere al dezvoltării ulterioare, pot fi luate in considerare mai multe aspecte:

1. Adaugarea orei in afisarea pe display (XX:XX:XX). Astfel, timpul afisat poate fi folosit ca un ceas uzual.

2. Adaugarea unui buton, utilizat numai pentru resetare, pentru simplificarea proiectului si a modului de functionare.

3. Adaugarea unui buton utilizat pentru incrementarea orelor, daca este realizata dezvoltarea de la subpunctul 1.

4. Adaugarea unei alarme cand numararea este crescatoare. De exemplu, daca se doreste aprinderea becului dupa 99 de minute sa nu fie nevoie ca utilizatorul sa apese butonul de incrementare a minutelor de 99 de ori, doar sa astepte dupa ce apasa START/STOP pana ajunge numaratoarea la 99 de minute.