MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Facultatea de Automatică și Calculatoare

Secția Calculatoare

Anul I

Grupa 30216

2016-2017

Reclamă Publicitară Cu Animații Multiple

Profesor coordonator:

Conf. Dr. Ing. Lucia Văcariu

Student:

Florea Gabriel-Alin

Cuprins

1.) Specificația proiectului 3
   a. Enunț 3
   b. Prezentare 4
2.) Schema bloc 7
3.) Descriere 8
   a. Funcționare 8
   b. Componente utilizate (cod și explicații) 9
4.) Justificarea soluției alese 29
5.) Instrucțiuni de utilizare și întreținere 30
6.) Poze cu funcționarea 36
7.) Posibilități de dezvoltare 38

Enunțul problemei

Să se proiecteze o reclamă publicitară cu animații multiple.

Se vor folosi afișajele cu 7 segmente.

Textul de afișat va fi format din simboluri ale unui alfabet disponibil.

Reclama va avea mai multe regimuri de funcționare (minimum 4) ce vor putea fi selectate de către utilizator, de la comutatoarele plăcuței cu FPGA.

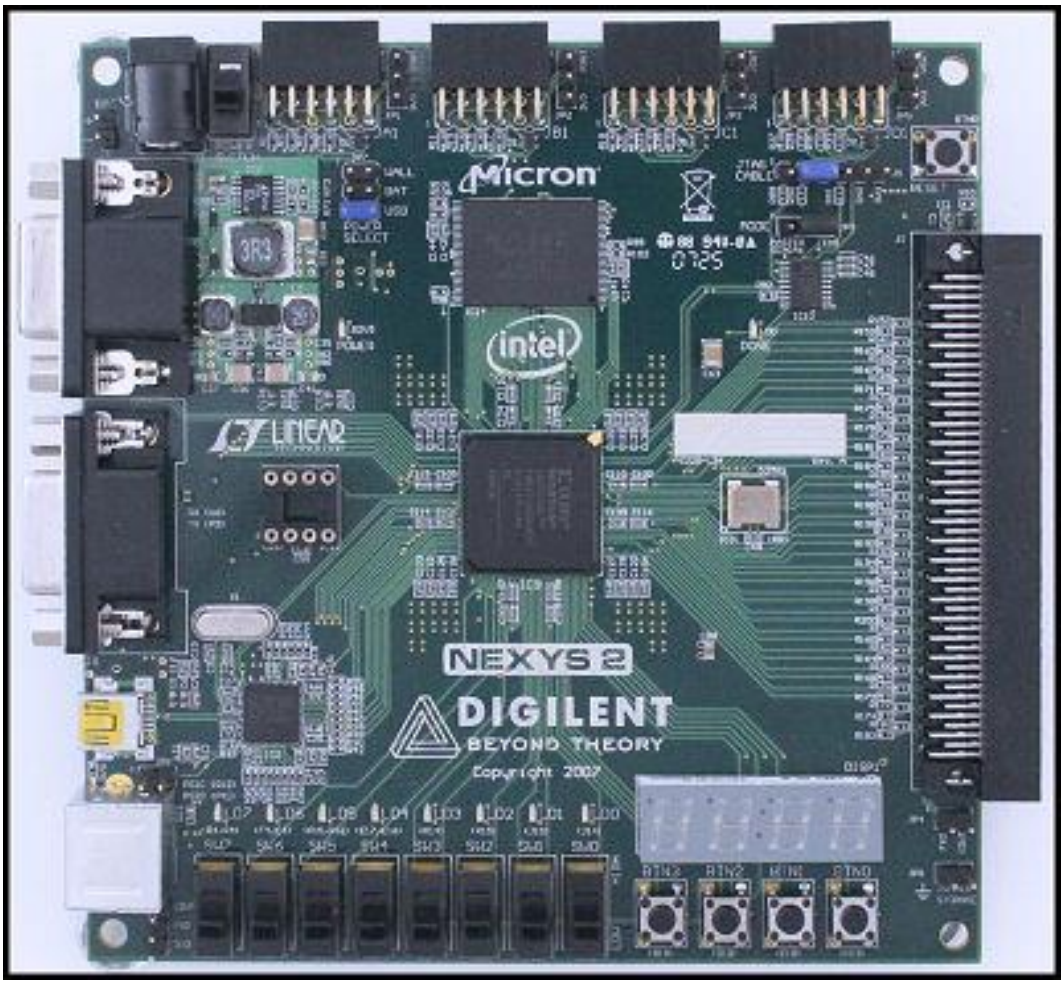
Se va folosi oscilatorul de cuarț încorporat în plăcuța cu FPGA (semnalul de clock respectiv va trebui desigur să fie divizat).

Exemple de regimuri de functionare: „curgerea” scrisului de la dreapta spre stânga, pâlpâire, afișaj literă cu literă etc.

Deoarece pe un afișaj cu 7 segmente nu se pot reprezenta toate literele, se va crea un alfabet maximal și mesajele vor fi compuse din simbolurile acelui alfabet. Mesajul va fi conținut într-o memorie pentru a putea fi ușor schimbat.

Proiectul va fi realizat de 1 student.

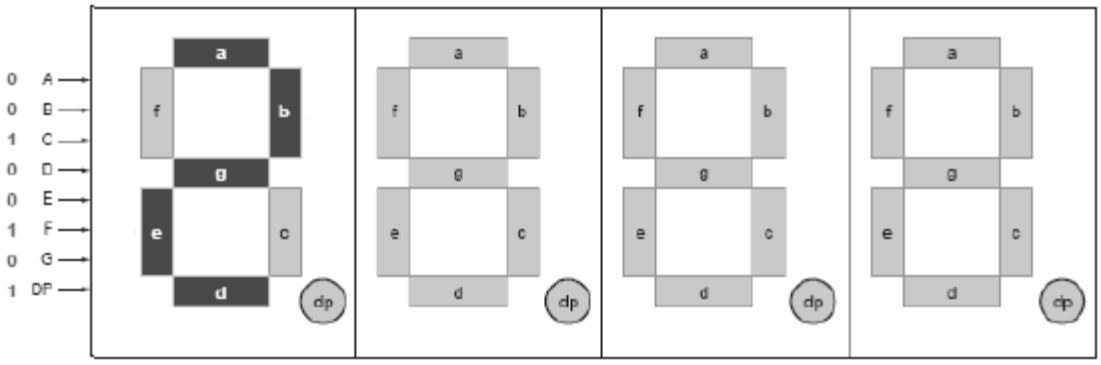
Prezentare placă NEXYS-2

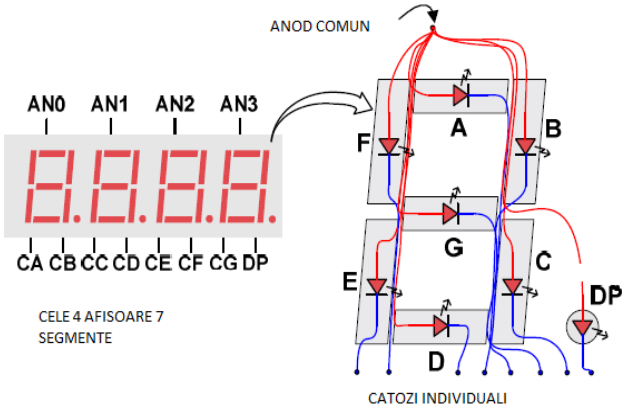


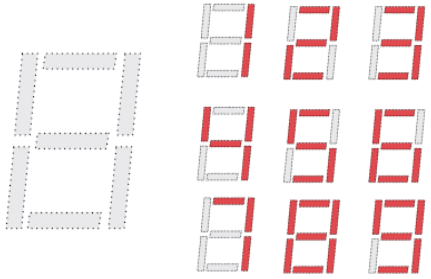
Placa cu FPGA NEXYS-2 este o placa complet funcțională ce face parte din seria Spartan 3E.

Placa cuprinde: atât componente standard (4 afișoare 7 segmente, 8 comutatoare cu 2 stări, 4 comutatoare de tip push-button, 8 leduri), cât și alte porturi de legătura cu exteriorul (interfețe de input/output). Oscilatorul de cuarț integrat al plăcii are frecvența de 50 MHz.

Afișoarele 7 segmente







Fiecare din cele 4 afișoare au în componență 7 segmente si un punct zecimal. Fiecare din cele 4 afișoare sunt activate de către un anod individual. Comanda de activare a segmentelor este comună tuturor celor 4 afișoare. Pentru a avea conținut diferit pe toate cele 4 afișoare, trebuie să schimbăm alternativ afișorul care este activat (prin anod) și conținutul segmentelor cu o frecvența care nu este perceptibila de către om. Astfel vom vedea pe toate afișoarele conținutul dorit.

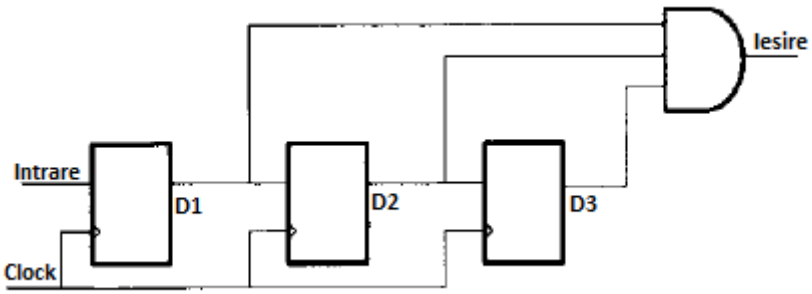
Atât segmentele, cât și anodurile sunt active pe 0 logic.

Deoarece pe un afișor 7 segmente nu se pot reprezenta toate caracterele, se va crea un dicționar de simboluri care vor fi utilizate.

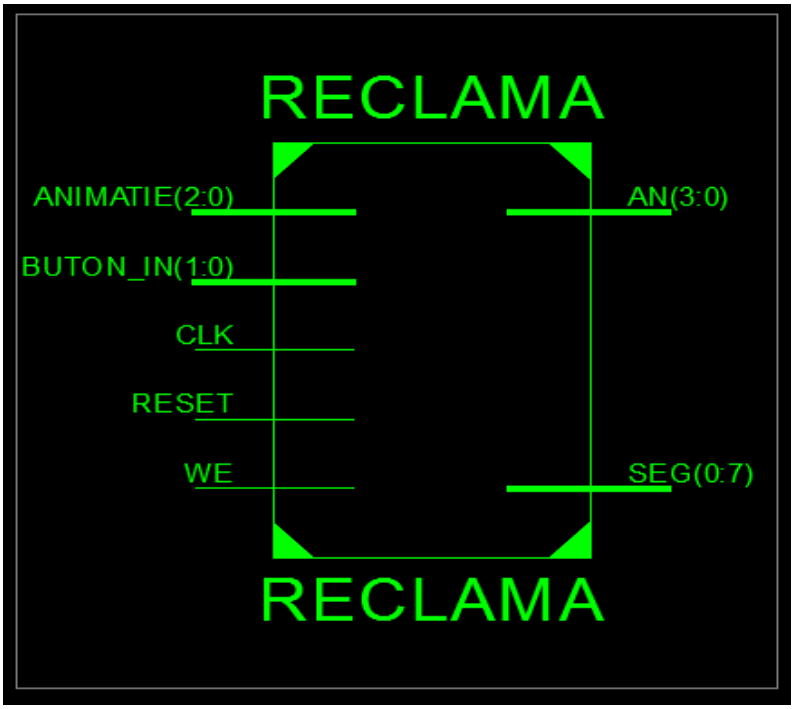
Fenomenul de bounce al butoanelor

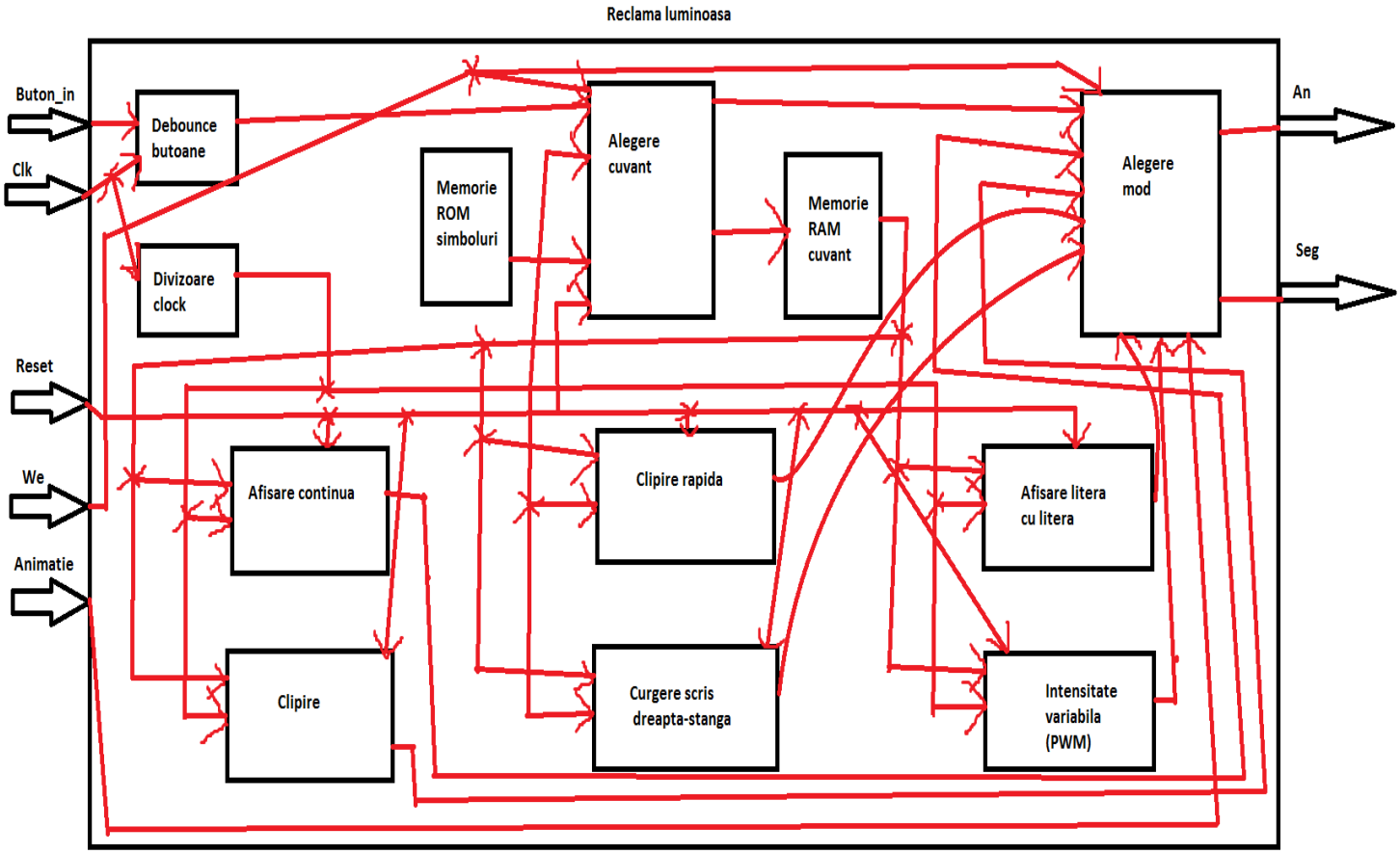
Datorită imperfecțiunilor mecanice ale contactelor electrice la închiderea sau la deschiderea elementelor apare un fenomen de oscilație a semnalului care poate conduce la citirea eronată a comenzii (citire multiplă).

Acest lucru impune realizarea unui circuit de debounce pentru butoane, pentru a putea stabiliza semnalul de comandă.



Schema bloc a proiectului





Modul de funcționare

La pornire, pe afișoare nu se afișeaza nimic.

Pentru a introduce cuvântul, care se dorește a fi afișat se activeaza intrarea WE de la switch-ul 0. Cu ajutorul butonului 3 se parcurge memoria de simboluri, iar cu ajutorul butonului 0 se confirmă acel simbol. După confirmarea celor 4 simboluri, se dezactivează intrarea WE.

De la switch-urile 7, 6 și 5 se selecteaza animația dorita:

1.) Pentru intrările „000”, „110”, sau „111” se activează modul „AFIȘARE CONTINUĂ”.
2.) Pentru intrările „001” se activează modul „CLIPIRE”.
3.) Pentru intrările „010” se activează modul „CLIPIRE RAPIDĂ”.
4.) Pentru intrările „011” se activează modul „CURGERE SCRIS”.
5.) Pentru intrările „100” se activează modul „AFIȘARE LITERĂ CU LITERĂ”.
6.) Pentru intrările „101” se activează modul „INTENSITATE VARIABILĂ”.

Dacă se dorește schimbarea cuvântului de afișat, se activează intrarea de resetare de pe switch-ul 3. După dezactivarea acesteia, se poate reîncepe procedura de alegere a cuvântului.

Componentele utilizate

1.) DEBOUNCE – Debounce pentru cele 2 butoane utilizate.

Se realizează debounce-ul butoanelor, utilizând un numărător și 3 semnale de delay (cu rol de bistabile). Utilizarea numărătorului alaturi de semnalele de delay are rolul de a asigura cu 100% siguranța debounce-ului, evitându-se astfel orice interferențe de natura electrică. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DEBOUNCE IS
        PORT( CLK: IN STD_LOGIC; --CLOCK-UL PLACUTEI
                        BUTON_IN: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --BUTOANELE LA INTRARE
                        BUTON_OUT: OUT STD_LOGIC_VECTOR(1 DOWNTO 0)); --BUTOANELE DUPA DEBOUNCE
END ENTITY;

ARCHITECTURE ARCH_DEBOUNCE OF DEBOUNCE IS
SIGNAL COUNTER: INTEGER RANGE 0 TO 500000:=0; --NUMARATOR IN BUCLA 0 500000
--SE FOLOSESTE PENTRU A SPORI SIGURANTA DEBOUNCE-ULUI PENTRU ORICE TIP DE PLACUTA CU FPGA
SIGNAL D1,D2,D3: STD_LOGIC_VECTOR (1 DOWNTO 0); --SEMNALE PENTRU DELAY (ROL DE BISTABILE)

BEGIN

PROCESS(CLK)
BEGIN
        IF RISING_EDGE(CLK) THEN                --DACA AVEM FRONT CRESCATOR PE CLOCK
                IF COUNTER=499999 THEN          --DACA AM AJUNS LA 500000
                                COUNTER<=0;                             --RESETAM NUMARATORUL
                                D1<=BUTON_IN;                   --FACEM ATRIBUIRI PENTRU INTARZIERE
                                D2<=D1;
                                D3<=D2;
                ELSE    COUNTER<=COUNTER+1;     --DACA NU AM AJUNS LA 500000 INCREMENTAM NUMARATORUL
                END IF;
        END IF;
END PROCESS;
BUTON_OUT<=D1 AND D2 AND D3; --SEMNALUL PENTRU BUTOANELE DUPA DEBOUNCE
END ARCHITECTURE;
```

2.) DIVIZOARE_CLOCK - Divizoarele de frecvență pentru semnalul de clock.

Divizarea semnalului de clock este necesară pentru a putea percepe cu ochiul uman diversele moduri de afișare a cuvântului.

Datorită faptului că placa cu FPGA NEXYS-2 generează un semnal de clock cu frecvența de 50MHz (adică au loc 50000000 de schimbări pe semnalul de clock în interval de 1 secundă), acesta trebuie divizat. Așadar, pentru a obține un semnal de clock cu frecvența de 1 Hz, trebuie ca la 25000000 de fronturi crescătoare, să inducem o schimbare pe noul semnal de clock (cel de 1Hz).

Semnalul TEMPO are o frecvență mare pentru a putea schimba anodurile afișoarelor, atât de rapid încât ochiul uman să perceapă ca și cum afișarea ar fi continuă.

Semnalul CLK_OUT_PWM are tot o frecvență mare pentru a putea controla modificarea valorii intensității luminoase (duty cycle-ului).

Notațiile utilizate sunt explicate în codul componentei

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DIVIZOARE IS
        PORT (CLK: IN STD_LOGIC; --INTRAREA CLOCK-ULUI DE PE PLACA
                        CLK_OUT_1SEC: OUT STD_LOGIC; --CLOCK LA 1 SECUNDA PENTRU CLIPIRE
                        CLK_OUT_05SEC: OUT STD_LOGIC; --CLOCK LA 0.5 SECUNDE PENTRU CLIPIRE RAPIDA
                        TEMPO: OUT STD_LOGIC_VECTOR( 1 DOWNTO 0); --CLOCK PENTRU SCHIMBAREA ANODURILOR LA AFISARE
                        CLK_OUT_DEPLASARE: OUT STD_LOGIC; --CLOCK PENTRU DEPLASARE, MAI MIC DE 1 SECUNDA
                        COUNTER1: OUT INTEGER; --NUMARATOR IN BUCLA 0-2300 PENTRU PWM
                        CLK_OUT_PWM: OUT STD_LOGIC); --CLOCK PENTRU MODIFICAREA INTENSITATII
END ENTITY;

ARCHITECTURE ARCH_DIVIZOARE OF DIVIZOARE IS
SIGNAL COUNTER_DIV: INTEGER RANGE 0 TO 25000000:=0; --NUMARATOR IN BUCLA 0 - 25000000 (1 SECUNDA)
SIGNAL COUNTER_DIV_RAPID: INTEGER RANGE 0 TO 12500000:=0; --NUMARATOR IN BUCLA 0 - 12500000 (0.5 SECUNDE)
SIGNAL COUNTER_DIV_DEPLASARE: INTEGER RANGE 0 TO 18000000:=0; --NUMARATOR IN BUCLA 0 - 18000000 (MAI PUTIN DE 1 SECUNDA)
SIGNAL T: STD_LOGIC:='0'; --SEMNAL DE CLOCK PENTRU 1 SECUNDA
SIGNAL T_PWM: STD_LOGIC:='0'; --SEMNAL DE CLOCK PENTRU PWM
SIGNAL S: STD_LOGIC:='0'; --SEMNAL DE CLOCK PENTRU MAI PUTIN DE 1 SECUNDA
SIGNAL T5: STD_LOGIC:='0'; --SEMNAL DE CLOCK PENTRU 0.5 SECUNDE
SIGNAL COUNTER_COMUT: INTEGER RANGE 0 TO 250000:=0; --NUMARATOR IN BUCLA 0 - 250000 (PENTRU SCHIMBAREA ANODURILOR)
SIGNAL TEMPORAR: STD_LOGIC_VECTOR(1 DOWNTO 0):="00"; --SEMNAL DE CLOCK PENTRU SCHIMBAREA ANODURILOR
SIGNAL COUNTER: INTEGER RANGE 0 TO 2300:=0; --NUMARATOR IN BUCLA 0 - 2300 PENTRU PWM
SIGNAL COUNTER2: INTEGER RANGE 0 TO 10500:=0; -- NUMARATOR IN BUCLA 0 - 10500 PENTRU PWM

BEGIN

PROCESS(CLK)
BEGIN
        IF RISING_EDGE(CLK) THEN  --DACA AVEM FRONT CRESCATOR PE CLOCK-UL PLACUTEI
```

```
            --DIVIZOR LA 1 SECUNDA
            IF COUNTER_DIV=24999999 THEN    --CAND AM AJUNS LA CAPATUL BUCLEI DE NUMARARE
                        COUNTER_DIV<=0;                 --RESETEZ NUMARATORUL
                        T<=NOT T;                                       --MODIFIC VALOAREA NOULUI SEMNAL DE CLOCK
            ELSE    COUNTER_DIV<=COUNTER_DIV+1; --DACA NU AM AJUNS LA CAPATUL BUCLEI DE NUMARARE INCREMENTEZ NUMARATORUL
            END IF;

            --DIVIZORUL LA 0.5 SECUNDE
            IF COUNTER_DIV_RAPID=12499999 THEN
                        COUNTER_DIV_RAPID<=0;
                        T5<=NOT T5;
            ELSE    COUNTER_DIV_RAPID<=COUNTER_DIV_RAPID+1;
            END IF;

            --DIVIZORUL PENTRU SCHIMBAREA ANODURILOR
            IF COUNTER_COMUT=249999 THEN
                        COUNTER_COMUT<=0;
                        TEMPORAR<=TEMPORAR+'1';
            ELSE    COUNTER_COMUT<=COUNTER_COMUT+1;
            END IF;

            --DIVIZORUL PENTRU CURGEREA SCRISULUI
            IF COUNTER_DIV_DEPLASARE=17999999 THEN
                        COUNTER_DIV_DEPLASARE<=0;
                        S<=NOT S;
            ELSE    COUNTER_DIV_DEPLASARE<=COUNTER_DIV_DEPLASARE+1;
            END IF;

            --NUMARATOR PENTRU PWM
            IF COUNTER=2299 THEN            --DACA AM AJUNS LA CAPATUL BUCLEI
                        COUNTER<=0;                             --RESETEZ NUMARATORUL
            ELSE    COUNTER<=COUNTER+1;     --DACA NU AM AJUNS LA CAPATUL BUCLEI DE NUMARARE INCREMENTEZ NUMARATORUL
            END IF;



             --DIVIZOR PENTRU PWM
             IF COUNTER2=10499 THEN
                          COUNTER2<=0;
                          T_PWM<=NOT T_PWM;
             ELSE    COUNTER2<=COUNTER2+1;
             END IF;
        END IF;
END PROCESS;

--ATRIBUIREA SEMNALELOR DE IESIRE DIN COMPONENTA CU VALORILE CORESPUNZATOARE
CLK_OUT_1SEC<=T;
CLK_OUT_05SEC<=T5;
TEMPO<=TEMPORAR;
CLK_OUT_DEPLASARE<=S;
COUNTER1<=COUNTER;
CLK_OUT_PWM<=T_PWM;

END ARCHITECTURE;
```

3.) ALEGEREA_CUVANTULUI – Alegerea cuvântului care se dorește a fi afișat.

Pentru alegerea cuvântului trebuie să fie activă intrarea de WE (write enable). Cu unul dintre butoane se parcurge memoria de simboluri, iar cu celălalt se confirmă alegerea acelui simbol de a face parte din cuvântul de afișat. Se pot confirma pentru afișare 0, 1, 2, 3, sau 4 simboluri, afișarea făcându-se corespunzător în fiecare caz. În momentul validării unui simbol, acesta se introduce în memoria RAM (pentru memorarea cuvântului). La fiecare pas, se afișează pe afișoarele 7 segmente ce s-a ales și simbolul în curs de alegere. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY ALEGERE IS
        PORT( WE: IN STD_LOGIC; --INTRAREA DE ACTIVARE A SCRIERII
                        RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        BUTON_OUT: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --INTRAREA BUTOANELOR (DUPA DEBOUNCE)
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE:        INOUT MEMORIE2; --MEMORIA RAM PENTRU CUVANTUL DE AFISAT
                        SIMBOL: IN MEMORIE1); --MEMORIA ROM PENTRU SIMBOLURI
END ENTITY;

ARCHITECTURE ARCH_ALEGERE OF ALEGERE IS
SIGNAL CONT: INTEGER RANGE 0 TO 4:=0; --SEMNAL PENTRU ALEGEREA CUVANTULUI

BEGIN

PROCESS(WE, RESET, TEMPO, CONT, CUVINTE)
VARIABLE X: INTEGER RANGE 0 TO 31:=0; --VARIABILA PENTRU PARCURGEREA MEMORIEI DE SIMBOLURI
BEGIN
--ALEGEREA LITERELOR DIN CUVANT
        IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
```

```
        IF CONT=0 THEN          --DACA NU AM CONFIRMAT NICI O LITERA
                        AN<="1110";     --AFISAM ULTIMUL AFISOR
                        SEG<=SIMBOL(X);--LITERA LA CARE SUNTEM IN MEMORIE
        ELSIF CONT=1 THEN --DACA AM CONFIRMAT O LITERA
                                IF TEMPO="00" OR TEMPO="10" THEN --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                        AN<="1110"; --AFISAM PE ULTIMUL AFISOR
                                        SEG<=SIMBOL(X); --LITERA LA CARE SUNTEM IN MEMORIE
                                ELSE    AN<="1101"; --PE PENULTIMUL AFISOR
                                        SEG<=CUVINTE(3); --AFISAM LITERA CONFIRMATA
                                END IF;
        ELSIF CONT=2 THEN --DACA AM CONFIRMAT 2 LITERE
                                IF TEMPO="00" OR TEMPO="10" THEN --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                        AN<="1110"; --AFISAM ULTIMUL AFISOR
                                        SEG<=SIMBOL(X); --LITERA LA CARE SUNTEM IN MEMORIE
                                ELSIF TEMPO="01" THEN
                                        AN<="1101"; --PE PENULTIMUL AFISOR
                                        SEG<=CUVINTE(2); --A DOUA LITERA ALEASA
                                ELSE    AN<="1011"; --PE AL DOILEA AFISOR
                                        SEG<=CUVINTE(3); --PRIMA LITERA ALEASA
                                END IF;
        ELSIF CONT=3 THEN  --DACA AM CONFIRMAT 3 LITERE
                                IF TEMPO="00" THEN --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                        AN<="1110"; --PE ULTIMUL AFISOR
                                        SEG<=SIMBOL(X); --LITERA LA CARE SUNTEM IN MEMORIE
                                ELSIF TEMPO="01" THEN
                                        AN<="1101"; --PE PENULTIMUL AFISOR
                                        SEG<=CUVINTE(1); -- A TREIA LITERA ALEASA
                                ELSIF TEMPO="10" THEN
                                        AN<="1011"; --PE AL DOILEA AFISOR
                                        SEG<=CUVINTE(2); --A DOUA LITERA ALEASA
                                ELSE    AN<="0111"; --PE PRIMUL AFISOR
                                        SEG<=CUVINTE(3); --PRIMA LITERA ALEASA
                                END IF;

        ELSIF CONT=4 THEN --DACA AM CONFIRMAT TOATE CELE 4 LITERE ALE CUVANTULUI
                                IF TEMPO="00" THEN
                                        AN<="1110"; --PE ULTIMUL AFISOR
                                        SEG<=CUVINTE(0); --A PATRA LITERA ALEASA
                                ELSIF TEMPO="01" THEN
                                        AN<="1101"; --PE PENULTIMUL AFISOR
                                        SEG<=CUVINTE(1); -- A TREIA LITERA ALEASA
                                ELSIF TEMPO="10" THEN
                                        AN<="1011"; --PE AL DOILEA AFISOR
                                        SEG<=CUVINTE(2); --A DOUA LITERA ALEASA
                                ELSE    AN<="0111"; --PE PRIMUL AFISOR
                                        SEG<=CUVINTE(3); --PRIMA LITERA ALEASA
                                END IF;
        END IF;

IF RISING_EDGE(BUTON_OUT(1)) AND WE='1' THEN X:=X+1;
        --DACA APASAM BUTONUL DE PARCURGERE A MEMORIEI SI
        --ESTE ACTIVATA INTRODUCEREA CUVANTULUI
        --INCREMENTAM VARIABILA DE PARCURGERE A MEMORIEI
END IF;

IF RISING_EDGE(BUTON_OUT(0))AND WE='1' THEN
        --DACA APASAM BUTONUL DE VALIDARE A CARACTERULUI ALES SI
        --ESTE ACTIVATA INTRODUCEREA CUVANTULUI
                        IF CONT<4  THEN --DACA AM VALIDAT MAI PUTIN DE 4 CARACTERE
                                        CONT<=CONT+1; --INCREMENTAM NUMARUL DE CARACTERE VALIDATE
                                        CUVINTE(4-CONT-1)<=SIMBOL(X); --INTRODUCEM CARACTERUL IN MEMORIA RAM
                                                                      --(ADICA IN MEMORIA UNDE E PUS CUVANTUL INTRODUS)
                        END IF;
END IF;
```

```
        ELSE    --DACA E ACTIVAT SEMNALUL DE RESET, RESETAM IN VALORILE PREDEFINITE SEMNALELE SI VARIABILA
                        X:=0;
                        CONT<=0;
                        AN<="1111";
                        SEG<="11111111";
                        CUVINTE(0)<=SIMBOL(31);
                        CUVINTE(1)<=SIMBOL(31);
                        CUVINTE(2)<=SIMBOL(31);
                        CUVINTE(3)<=SIMBOL(31);
        END IF;
END PROCESS;
END ARCHITECTURE;
```

4.) AFISARE_CONTINUA – Afișare în mod continuu a cuvântului ales.

Când intrările pentru animație sunt: „000”, „110”, sau „111” se afișează cuvântul în mod continuu. Sunt 3 intrări pentru aceeași animație, deoarece proiectul conține 6 animații, iar pe 3 biți se pot obtine 8 combinții diferite. Astfel se elimină stările necunoscute și se evită problemele de hazard.

În funcție de semnalul de clock pentru modificarea anodurilor, se afișează pe fiecare afișor conținutul corespunzător. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY CONTINUU IS
       PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                     TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                     AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                     SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                     CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_CONTINUU OF CONTINUU IS

BEGIN
```

```
PROCESS(RESET, CUVINTE, TEMPO)
BEGIN
        IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
                 CASE TEMPO IS --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                                          --AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
                                                 WHEN "00" => AN<="1110"; SEG<=CUVINTE(0);
                                                 WHEN "01" => AN<="1101"; SEG<=CUVINTE(1);
                                                 WHEN "10" => AN<="1011"; SEG<=CUVINTE(2);
                                                 WHEN "11" => AN<="0111"; SEG<=CUVINTE(3);
                                                 WHEN OTHERS => AN<="1111";
                END CASE;
        ELSE    --DACA E ACTIVAT SEMNALUL DE RESETARE
                        AN<="1111";
                        SEG<="11111111";
        END IF;
END PROCESS;
END ARCHITECTURE;
```

5.) CLIPIRE – Clipire la 1 secunda.

Când intrările pentru animație sunt: „001”, textul reclamei clipește o dată pe secundă.

În momentul în care clock-ul de 1 secundă este pe 1 logic, afișăm conținutul corespunzător pe cele 4 afișoare în funcție de clock-ul de schimbare a anodurilor, iar atunci când este pe 0 logic nu afișăm nimic. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY CLIPIRE IS
       PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                     T: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA 1 SECUNDA
                     TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                     AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                     SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                     CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_CLIPIRE OF CLIPIRE IS

BEGIN
```

```
PROCESS(RESET, T, CUVINTE, TEMPO)
BEGIN
	IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
		IF T='1' THEN --CAND CLOCK-UL DE 1 SECUNDA ESTE PE 1 LOGIC
			 CASE TEMPO IS --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
								--AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
					WHEN "00" => AN<="1110"; SEG<=CUVINTE(0);
					WHEN "01" => AN<="1101"; SEG<=CUVINTE(1);
					WHEN "10" => AN<="1011"; SEG<=CUVINTE(2);
					WHEN "11" => AN<="0111"; SEG<=CUVINTE(3);
					WHEN OTHERS => AN<="1111";
			END CASE;
		ELSE --CAND CLOCK-UL DE 1 SECUNDA ESTE PE 0 LOGIC
			AN<="1111"; --NU AFISAM NIMIC
		END IF;
	ELSE	--DACA E ACTIVAT SEMNALUL DE RESETARE
		AN<="1111";
		SEG<="11111111";
	END IF;
END PROCESS;
END ARCHITECTURE;
```

6.) CLIPIRE_RAPIDA – Clipire la 0.5 secunde.

Când intrările pentru animație sunt: „010”, textul reclamei clipește de două ori pe secundă.

În momentul în care clock-ul de 0.5 secunde este pe 1 logic, afișăm conținutul corespunzător pe cele 4 afișoare în funcție de clock-ul de schimbare a anodurilor, iar atunci când este pe 0 logic nu afișăm nimic. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI
```

```
ENTITY CLIPIRERAPIDA IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        T5: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA 0.5 SECUNDE
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_CLIPIRERAPIDA OF CLIPIRERAPIDA IS

BEGIN

PROCESS(RESET, T5, CUVINTE, TEMPO)
BEGIN
        IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
                IF T5='1' THEN --CAND CLOCK-UL DE 0.5 SECUNDE ESTE PE 1 LOGIC
                         CASE TEMPO IS --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                                        --AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
                                                WHEN "00" => AN<="1110"; SEG<=CUVINTE(0);
                                                WHEN "01" => AN<="1101"; SEG<=CUVINTE(1);
                                                WHEN "10" => AN<="1011"; SEG<=CUVINTE(2);
                                                WHEN "11" => AN<="0111"; SEG<=CUVINTE(3);
                                                WHEN OTHERS => AN<="1111";
                        END CASE;
                ELSE  --CAND CLOCK-UL DE 0.5 SECUNDE ESTE PE 0 LOGIC
                        AN<="1111";
                END IF;
        ELSE --DACA E ACTIVAT SEMNALUL DE RESETARE
                AN<="1111";
                SEG<="11111111";
        END IF;
END PROCESS;
END ARCHITECTURE;
```

7.) CURGERE_SCRIS – Curgerea scrisului de la dreapta spre stânga (deplasarea acestuia).

Când intrările pentru animație sunt: „011”, textul reclamei curge de la dreapta spre stânga (se deplasează de la dreapta spre stânga).

Daca avem front crescător pe clock-ul mai mic de 1 secundă (0.72 secunde), atunci realizăm deplasarea literelor componente ale cuvântului. După ce a fost parcurs complet cuvântul, adăugăm un spațiu de demarcație.

Afișarea o realizăm în funcție de clock-ul de schimbare a anodurilor. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY CURGERE_SCRIS IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        S: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA MAI PUTIN DE 1 SECUNDA
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_CURGERE_SCRIS OF CURGERE_SCRIS IS
SIGNAL I: INTEGER RANGE 0 TO 4:=0; --INDICE PENTRU DEPLASARE
SIGNAL DATA1, DATA2, DATA3, DATA4 : STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEMNALE PENTRU DEPLASARE (CURGEREA SCRISULUI)

BEGIN

PROCESS(RESET, S, TEMPO, I)
BEGIN
IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
        IF RISING_EDGE(S) THEN --DACA AVEM FRONT CRESCATOR PE CLOCK-UL MAI MIC DE 1 SECUNDA
                --REALIZAM DEPLASAREA
                --DEPLASARE (SE FACE IN MOD CONCURENT);

                        IF I=4 THEN --DACA AM PARCURS COMPLET CUVANTUL, PUNEM UN SPATIU SI REALIZAM DEPLASAREA IN CONTINUARE
                                DATA1<="11111111";
                                DATA2<=DATA1;
                                DATA3<=DATA2;
                                DATA4<=DATA3;
                                I<=0;
                        ELSE --DACA NU AM PARCURS COMPLET CUVANTUL, REALIZAM DEPLASAREA IN CONTINUARE
                                DATA1<=CUVINTE(3-I);
                                DATA2<=DATA1;
                                DATA3<=DATA2;
                                DATA4<=DATA3;
                                I<=I+1;
                        END IF;
                END IF;

                        CASE TEMPO IS   --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                                        --AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
                                        WHEN "00" => AN<="1110"; SEG<=DATA1;
                                        WHEN "01" => AN<="1101"; SEG<=DATA2;
                                        WHEN "10" => AN<="1011"; SEG<=DATA3;
                                        WHEN "11" => AN<="0111"; SEG<=DATA4;
                                        WHEN OTHERS => AN<="1111";
                        END CASE;
        ELSE --DACA E ACTIVAT SEMNALUL DE RESETARE
                AN<="1111";
                SEG<="11111111";
                I<=0;
                DATA1<="11111111";
                DATA2<="11111111";
                DATA3<="11111111";
                DATA4<="11111111";
        END IF;
END PROCESS;
END ARCHITECTURE;
```

8.) LITERA_CU_LITERA – Afișarea cuvântului literă cu literă.

Când intrările pentru animație sunt: „100”, textul reclamei este afișat literă cu literă pe câte un afișor, în timp ce pe celelalte afișoare este afișată liniuță.

Atunci când avem front crescător pe clock-ul mai mic de 1 secundă (0.72 secunde) alegem afișorul care va afișa un caracter al cuvântului și cele care vor afișa liniuța.

Afișarea se face în funcție de clock-ul de schimbare a anodurilor.

Semnalul de resetare, funcționează similar ca și în celelalte componente. Atunci când este activat se resetează datele din această componentă la valorile inițiale. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY LITERA_CU_LITERA IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        S: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA MAI PUTIN DE 1 SECUNDA
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_LITERA_CU_LITERA OF LITERA_CU_LITERA IS
SIGNAL D1, D2, D3, D4 : STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEMNALE PENTRU AFISARE LITERA CU LITERA
SIGNAL I1: INTEGER RANGE 0 TO 3:=0; --INDICE PENTRU AFISARE LITERA CU LITERA

BEGIN
```

```
PROCESS(RESET, S, TEMPO, I1)
BEGIN
IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE
        IF RISING_EDGE(S) THEN --DACA AVEM FRONT CRESCATOR PE CLOCK-UL MAI MIC DE 1 SECUNDA
                                IF I1=0 THEN  --AFISEZ PRIMA LITERA PE PRIMUL AFISOR SI IN REST LINIUTE
                                        D1<=CUVINTE(3);
                                        D2<="11111101";
                                        D3<="11111101";
                                        D4<="11111101";
                                ELSIF I1=1 THEN --AFISEZ A DOUA LITERA PE AL DOILEA AFISOR SI IN REST LINIUTE
                                        D1<="11111101";
                                        D2<=CUVINTE(2);
                                        D3<="11111101";
                                        D4<="11111101";
                                ELSIF I1=2 THEN --AFISEZ A TREIA LITERA PE AL TREILEA AFISOR SI IN REST LINIUTE
                                        D1<="11111101";
                                        D2<="11111101";
                                        D3<=CUVINTE(1);
                                        D4<="11111101";
                                ELSIF I1=3 THEN --AFISEZ A PATRA LITERA PE AL PATRULEA AFISOR SI IN REST LINIUTE
                                        D1<="11111101";
                                        D2<="11111101";
                                        D3<="11111101";
                                        D4<=CUVINTE(0);
                                END IF;

                        IF I1<3 THEN --DACA NU AM AJUNS LA INDICELE 3 INCREMENTAM NUMARATORUL
                                        I1<=I1+1;
                        ELSE    I1<=0; --DACA AM AJUNS LA 3 RESETAM IN 0 NUMARATORUL
                        END IF;

        END IF;


                         CASE TEMPO IS --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
                                                        --AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
                                                        --AFISAREA O FACEM PUNAND DATELE IN ORDINE INVERSA
                                        WHEN "00" => AN<="1110"; SEG<=D4;
                                        WHEN "01" => AN<="1101"; SEG<=D3;
                                        WHEN "10" => AN<="1011"; SEG<=D2;
                                        WHEN "11" => AN<="0111"; SEG<=D1;
                                        WHEN OTHERS => AN<="1111";
                         END CASE;

ELSE --DACA E ACTIV SEMNALUL DE RESETARE
        AN<="1111";
        SEG<="11111111";
        I1<=0;
        D1<="11111111";
        D2<="11111111";
        D3<="11111111";
        D4<="11111111";
END IF;
END PROCESS;
END ARCHITECTURE;
```

9.) PWM – Afișarea cuvântului ales cu o intensitate care variază.

Când intrările pentru animație sunt: „101”, textul reclamei este afișat cu o intensitate variablilă, care variază între 0 și maxim și înapoi în 0 într-un interval de aproximativ 2 secunde.

Cât timp numărătorul pentru intensitate variabilă este mai mic decât valoarea duty cycle-ului, segmentele sunt active, iar când depașește această valoare sunt inactive. Datorită faptului că acest lucru se întâmplă foarte rapid, segmentele vor aparea cu o anumita intensitate.

Valoarea duty cycle-ului se modifică în funcție de clock-ul de modificare și în funcție de un semnal de ok, care comandă dacă să crească sau să scadă. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--PACHET PENTRU DECLARAREA TIPURILOR DE MEMORIE UTILIZATA
PACKAGE TIPURI IS
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA ROM PENTRU SIMBOLURI
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA RAM PENTRU CUVANTUL INTRODUS
END PACKAGE;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE TIPURI.ALL; --UTILIZAREA PACHETULUI

ENTITY PWM IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        T1: IN STD_LOGIC; --CLOCK PENTRU MODIFICAREA INTENSITATII
                        COUNTER1: IN INTEGER RANGE 0 TO 2300; --NUMARATOR IN BUCLA 0-2300 PENTRU PWM
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END ENTITY;

ARCHITECTURE ARCH_PWM OF PWM IS
SIGNAL C1, C2, C3, C4 : STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEMNALE PENTRU MEMORAREA DATELOR PENTRU PWM
--SUNT NECESARE PENTRU A PUTEA VEDEA INTENSITATI DIFERITE
SIGNAL VALOARE: INTEGER RANGE 0 TO 2300:=0; --VALOARE MAXIMA PENTRU PWM
SIGNAL OK: STD_LOGIC; --SEMNAL CARE CONDITIONEAZA CRESTEREA SAU SCADEREA INTENSITATII PENTRU PWM

BEGIN
```

```
PROCESS(RESET, T1, COUNTER1, TEMPO, OK, VALOARE)
BEGIN
IF RESET='0' THEN --DACA NU E ACTIVAT SEMNALUL DE RESETARE

	IF COUNTER1<=VALOARE THEN	--DACA NUMARATORUL E MAI MIC SAU EGAL DECAT VALOAREA MAXIMA
									--AFISEZ PE AFISOARE CONTINUTUL CORESPUNZATOR
		C1<=CUVINTE(0);
		C2<=CUVINTE(1);
		C3<=CUVINTE(2);
		C4<=CUVINTE(3);
		ELSE	--DACA NUMARATORUL E MAI MIC SAU EGAL DECAT VALOAREA MAXIMA
				--AFISEZ PE AFISOARE CONTINUTUL CORESPUNZATOR
		C1<="11111111";
		C2<="11111111";
		C3<="11111111";
		C4<="11111111";
	END IF;

	IF RISING_EDGE(T1) THEN --DACA AM FRONT CRESCATOR PE CLOCK-UL DE MODIFICARE A VALORII PENTRU PWM
		IF OK='0' AND VALOARE<2299 THEN VALOARE<=VALOARE+1; --DACA OK E 0 LOGIC SI VALOAREA E MAI MICA DECAT 2299 INCREMENTEZ VALOAREA
		ELSIF OK='0' AND VALOARE>=2299 THEN OK<='1'; VALOARE<=VALOARE-1; --DACA OK E 0 LOGIC SI VALOAREA E MAI MARE DECAT 2299 DECREMENTEZ VALOAREA
		ELSIF OK='1' AND VALOARE>0 THEN VALOARE<=VALOARE-1; --DACA OK E 1 LOGIC SI VALOAREA E MAI MARE DECAT 0 DECREMENTEZ VALOAREA
		ELSIF OK='1' AND VALOARE=0 THEN OK<='0'; VALOARE<=VALOARE+1; --DACA OK E 1 LOGIC SI VALOAREA E EGALA CU 0 INCREMENTEZ VALOAREA
		END IF;
	END IF;

		 CASE TEMPO IS --IN FUNCTIE DE CLOCK-UL DE SCHIMBARE A ANODURILOR
					--AFISAM PE FIECARE ANOD CONTINUTUL CORESPUNZATOR
				WHEN "00" => AN<="1110"; SEG<=C1;
				WHEN "01" => AN<="1101"; SEG<=C2;
				WHEN "10" => AN<="1011"; SEG<=C3;
				WHEN "11" => AN<="0111"; SEG<=C4;
				WHEN OTHERS => AN<="1111";
		END CASE;

ELSE --DACA E ACTIV SEMNALUL DE RESETARE
	 AN<="1111";
	 SEG<="11111111";
	 VALOARE<=0;
	 OK<='0';
	 C1<="11111111";
	 C2<="11111111";
	 C3<="11111111";
	 C4<="11111111";
END IF;
END PROCESS;
END ARCHITECTURE;
```

10.) ALEGERE_MOD – Alegerea modului de afișare, în funcție de animația aleasă.

Se aleg segmentele și anodurile care se afișează, în funcție de codul animației alese și de WE (write enable). Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY ALEGERE_MOD IS
	PORT ( WE: IN STD_LOGIC; --ALEGEREA INTRE INTRODUCEREA CUVANTULUI SI AFISAREA ACESTUIA
			SEG1, SEG2, SEG3, SEG4, SEG5, SEG6, SEG7: IN STD_LOGIC_VECTOR(0 TO 7); --SEGMENTELE PENTRU TOATE MODURILE DE AFISARE
			AN1, AN2, AN3, AN4, AN5, AN6, AN7: IN STD_LOGIC_VECTOR(3 DOWNTO 0); --ANODURILE PENTRU TOATE MODURILE DE AFISARE
			ANIMATIE: IN STD_LOGIC_VECTOR(2 DOWNTO 0); --ALEGEREA ANIMATIEI (MODULUI DE AFISARE)
			AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
			SEG: OUT STD_LOGIC_VECTOR(0 TO 7)); --IESIREA DE SEGMENTE
END ENTITY;

ARCHITECTURE ARCH_ALEGERE_MOD OF ALEGERE_MOD IS

BEGIN

PROCESS(WE, ANIMATIE, SEG1, SEG2, SEG3, SEG4, SEG5, SEG6, SEG7, AN1, AN2, AN3, AN4, AN5, AN6, AN7)
BEGIN
	IF WE='1'  THEN SEG<=SEG1; AN<=AN1; --DACA WE E PORNIT AFISAM CARACTERELE LA INTRODUCEREA CUVANTULUI
	ELSIF WE='0' AND (ANIMATIE="000" OR ANIMATIE="110" OR ANIMATIE="111") THEN SEG<=SEG2; AN<=AN2; --AFISARE NORMALA (APRINS CONTINUU)
	ELSIF WE='0' AND ANIMATIE="001" THEN SEG<=SEG3; AN<=AN3; --CLIPIRE
	ELSIF WE='0' AND ANIMATIE="010" THEN SEG<=SEG4; AN<=AN4;		--CLIPIRE RAPIDA
	ELSIF WE='0' AND ANIMATIE="011" THEN SEG<=SEG5; AN<=AN5;		--CURGERE SCRIS
	ELSIF WE='0' AND ANIMATIE="100" THEN SEG<=SEG6; AN<=AN6;		--LITERA CU LITERA
	ELSIF WE='0' AND ANIMATIE="101" THEN SEG<=SEG7; AN<=AN7;		--PWM
	ELSE AN<="1111"; SEG<="11111111"; --IN ALT CAZ NU AFISAM NIMIC
	END IF;
END PROCESS;
END ARCHITECTURE;
```

11.) MAIN – Modulul principal al programului, în care sunt instanțiate celelalte componente.

Acesta este modulul principal, care instanțiază toate celelalte componente, pentru a putea face reclama luminoasă să funcționeze corespunzător. Tot aici, sunt declarate și inițializate memoriile utilizate (memoria ROM pentru simboluri, și memoria RAM pentru cuvântul de afișat). În memorii se memorează, chiar codul pentru afișorul 7 segmente corespunzător caracterului. Notațiile utilizate sunt explicate în codul componentei.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY RECLAMA IS
        PORT( BUTON_IN: IN STD_LOGIC_VECTOR(1 DOWNTO 0);        --CELE 2 BUTOANE PENTRU ALEGERA CUVANTULUI
                        WE: IN STD_LOGIC; --ALEGEREA INTRE INTRODUCEREA CUVANTULUI SI AFISAREA ACESTUIA
                        ANIMATIE: IN STD_LOGIC_VECTOR(2 DOWNTO 0); --ALEGEREA ANIMATIEI
                        CLK: IN STD_LOGIC; --CLOCK-UL DE LA PLACUTA
                        RESET: IN STD_LOGIC; --SEMNAL DE RESETARE
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA PENTRU ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7)); --IESIREA PENTRU SEGMENTE
                        --IN TOT PROIECTUL AM FOLOSIT PENTRU AFISOARELE 7 SEGMENTE STD_LOGIC_VECTOR(0 TO 7)
                        --PENTRU A PUTEA CONTROLA SI PUNCTUL ZECIMAL DACA S-AR DORI ACEST LUCRU
END ENTITY;

ARCHITECTURE ARCH_RECLAMA OF RECLAMA IS


SIGNAL BUTON_OUT: STD_LOGIC_VECTOR(1 DOWNTO 0); --BUTOANELE DUPA DEBOUNCE
SIGNAL T: STD_LOGIC; --NOUL CLOCK CU FRECVENTA DE 1 SECUNDA
SIGNAL T5: STD_LOGIC; --NOUL CLOCK CU FRECVENTA DE 0.5 SECUNDE
SIGNAL S: STD_LOGIC; --CLOCK DIVIZAT PENTRU REGISTRU
SIGNAL TEMPO: STD_LOGIC_VECTOR(1 DOWNTO 0);      --PENTRU SCHIMBARE ANODURILOR
SIGNAL COUNTER1: INTEGER RANGE 0 TO 2300; --PENTRU PWM
SIGNAL T1: STD_LOGIC; --PENTRU PWM
SIGNAL AN1: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU ALEGEREA CVANTULUI
SIGNAL SEG1: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU ALEGEREA CUVANTULUI
SIGNAL AN2: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU AFISARE CONTINUA
SIGNAL SEG2: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU AFISARE CONTINUA
SIGNAL AN3: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; -- ANODURILE PENTRU CLIPIRE LA 1 SECUNDA
SIGNAL SEG3: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU CLIPIRE LA 1 SECUNDA
SIGNAL AN4: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU CLIPIRE RAPIDA
SIGNAL SEG4: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU CLIPIRE RAPIDA
SIGNAL AN5: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU CURGERE SCRIS
SIGNAL SEG5: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU CURGERE SCRIS
SIGNAL AN6: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU AFISARE LITERA CU LITERA
SIGNAL SEG6: STD_LOGIC_VECTOR(0 TO 7):="11111111"; --SEGMENTELE PENTRU AFISARE LITERA CU LITERA
SIGNAL AN7: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111"; --ANODURILE PENTRU INTENSITATE VARIABILA
SIGNAL SEG7: STD_LOGIC_VECTOR(0 TO 7):="11111111";      --SEGMENTELE PENTRU INTENSITATE VARIABILA

--TIPURILE DE MEMORIE
TYPE MEMORIE1 IS ARRAY (31 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA DE SIMBOLURI - O FOLOSIM CA MEMORIE ROM
TYPE MEMORIE2 IS ARRAY (3 DOWNTO 0) OF STD_LOGIC_VECTOR(0 TO 7); --MEMORIA PENTRU CUVANT - O FOLOSIM CA MEMORIE RAM

--COMPONENTA CE DIVIZEAZA CLOCK-UL LA DIFERITE FRECVENTE
COMPONENT DIVIZOARE IS
        PORT (CLK: IN STD_LOGIC; --INTRAREA CLOCK-ULUI DE PE PLACA
                        CLK_OUT_1SEC: OUT STD_LOGIC; --CLOCK LA 1 SECUNDA PENTRU CLIPIRE
                        CLK_OUT_05SEC: OUT STD_LOGIC; --CLOCK LA 0.5 SECUNDE PENTRU CLIPIRE RAPIDA
                        TEMPO: OUT STD_LOGIC_VECTOR( 1 DOWNTO 0); --CLOCK PENTRU SCHIMBAREA ANODURILOR LA AFISARE
                        CLK_OUT_DEPLASARE: OUT STD_LOGIC; --CLOCK PENTRU DEPLASARE, MAI MIC DE 1 SECUNDA
```

```
END COMPONENT;

--COMPONENTA DE DEBOUNCE DE BUTOANE
COMPONENT DEBOUNCE IS
PORT (CLK: IN STD_LOGIC; --INTRAREA CLOCK-ULUI DE PE PLACA
               BUTON_IN : IN STD_LOGIC_VECTOR(1 DOWNTO 0); --BUTOANELE LA INTRARE
               BUTON_OUT : OUT STD_LOGIC_VECTOR(1 DOWNTO 0)); --BUTOANELE DUPA DEBOUNCE
END COMPONENT;

--COMPONENTA DE ALEGERE A CUVANTULUI DE AFISAT
COMPONENT ALEGERE IS
        PORT( WE: IN STD_LOGIC; --INTRAREA DE ACTIVARE A SCRIERII
                        RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        BUTON_OUT: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --INTRAREA BUTOANELOR (DUPA DEBOUNCE)
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE:        INOUT MEMORIE2; --MEMORIA RAM PENTRU CUVANTUL DE AFISAT
                        SIMBOL: IN MEMORIE1); --MEMORIA ROM PENTRU SIMBOLURI
END COMPONENT;

--COMPONENTA PENTRU AFISARE CONTINUA
COMPONENT CONTINUU IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;


--COMPONENTA PENTRU CLIPIRE
COMPONENT CLIPIRE IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        T: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA 1 SECUNDA
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;

--COMPONENTA PENTRU CLIPIRE RAPIDA
COMPONENT CLIPIRERAPIDA IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        T5: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA 0.5 SECUNDE
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;

--COMPONENTA PENTRU CURGEREA SCRISULUI DE LA DREAPTA SPRE STANGA
COMPONENT CURGERE_SCRIS IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        S: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA MAI PUTIN DE 1 SECUNDA
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;
```

```
--COMPONENTA PENTRU AFISARE LITERA CU LITERA
COMPONENT LITERA_CU_LITERA IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        S: IN STD_LOGIC; --CLOCK-UL DIVIZAT LA MAI PUTIN DE 1 SECUNDA
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;

--COMPONENTA PENTRU INTENSITATE VARIABILA
COMPONENT PWM IS
        PORT( RESET: IN STD_LOGIC; --INTRAREA DE RESETARE
                        T1: IN STD_LOGIC; --CLOCK PENTRU MODIFICAREA INTENSITATII
                        COUNTER1: IN INTEGER RANGE 0 TO 2300; --NUMARATOR IN BUCLA 0-2300 PENTRU PWM
                        TEMPO: IN STD_LOGIC_VECTOR(1 DOWNTO 0); --CLOCK-UL PENTRU SCHIMBAREA ANODURILOR
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7); --IESIREA DE SEGMENTE
                        CUVINTE: IN MEMORIE2); -- MEMORIA PENTRU CUVANTUL DE AFISAT (O FOLOSIM CU MOD IN PENTRU CA NU AVEM DE SCRIS IN EA)
END COMPONENT;

--COMPONENTA PENTRU ALEGEREA MODULUI DE AFISARE
COMPONENT ALEGERE_MOD IS
        PORT ( WE: IN STD_LOGIC; --ALEGEREA INTRE INTRODUCEREA CUVANTULUI SI AFISAREA ACESTUIA
                        SEG1, SEG2, SEG3, SEG4, SEG5, SEG6, SEG7: IN STD_LOGIC_VECTOR(0 TO 7); --SEGMENTELE PENTRU TOATE MODURILE DE AFISARE
                        AN1, AN2, AN3, AN4, AN5, AN6, AN7: IN STD_LOGIC_VECTOR(3 DOWNTO 0); --ANODURILE PENTRU TOATE MODURILE DE AFISARE
                        ANIMATIE: IN STD_LOGIC_VECTOR(2 DOWNTO 0); --ALEGEREA ANIMATIEI (MODULUI DE AFISARE)
                        AN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --IESIREA DE ANODURI
                        SEG: OUT STD_LOGIC_VECTOR(0 TO 7)); --IESIREA DE SEGMENTE
END COMPONENT;

--MEMORIA DE SIMBOLURI
CONSTANT SIMBOL: MEMORIE1:=(
0=> "00000011",  --0
1=> "10011111",  --1
2=> "00100101",  --2
3=> "00001101",  --3
4=> "10011001",  --4
5=> "01001001",  --5
6=> "01000001",  --6
7=> "00011111",  --7
8=> "00000001",  --8
9=> "00001001",  --9
10=> "00010001", --A
11=> "11000001", --b
12=> "01100011", --C
13=> "10000101", --d
14=> "01100001", --E
15=> "01110001", --F
16=> "01000011", --G
17=> "10010001", --H
18=> "11110011", --I
19=> "10000111", --J
20=> "01010001", --k
21=> "11100011", --L
22=> "11010101", --n
23=> "11000101", --o
24=> "00110001", --P
25=> "00011001", --q
26=> "11110101", --r
27=> "01001001", --S
28=> "11100001", --t
29=> "10000011", --U
30=> "10001001", --y
31=> "11111111");--spatiu
```

```
--MEMORIA PENTRU CUVANT
SIGNAL CUVINTE: MEMORIE2:=(
0=> SIMBOL(31),
1=> SIMBOL(31),
2=> SIMBOL(31),
3=> SIMBOL(31));

BEGIN

--INSTANTIEREA COMPONENTELOR:

--DEBOUNCE BUTOANE
DEBOUNCEBUTOANE: DEBOUNCE PORT MAP (CLK, BUTON_IN, BUTON_OUT);

--DIVIZOARELE
DIVIZOARE_CLOCK: DIVIZOARE PORT MAP (CLK, T, T5, TEMPO, S, COUNTER1, T1);

--ALEGEREA CUVANTULUI
ALEGERECUVANT: ALEGERE PORT MAP (WE, RESET, TEMPO, BUTON_OUT, AN1, SEG1, CUVINTE, SIMBOL);

--AFISAREA EFECTIVA

        --AFISAREA CONTINUA
        AFISARECONTINUA: CONTINUU PORT MAP (RESET, TEMPO, AN2, SEG2, CUVINTE);

        --CLIPIRE
        CLIPIRE1SEC: CLIPIRE PORT MAP (RESET, T, TEMPO, AN3, SEG3, CUVINTE);

        --CLIPIRE RAPIDA
        CLIPIRE05SEC: CLIPIRERAPIDA PORT MAP (RESET, T5, TEMPO, AN4, SEG4, CUVINTE);

        --CURGERE SCRIS
        CURGERESCRIS: CURGERE_SCRIS PORT MAP (RESET, S, TEMPO, AN5, SEG5, CUVINTE);

      --LITERA CU LITERA
      LITERACULITERA: LITERA_CU_LITERA PORT MAP (RESET, S, TEMPO, AN6, SEG6, CUVINTE);

      --INTENSITATE VARIABILA
      INTENSITATEVARIABILA: PWM PORT MAP (RESET, T1, COUNTER1, TEMPO, AN7, SEG7, CUVINTE);

      --ALEGERE MOD
      ALEGEREMOD: ALEGERE_MOD PORT MAP (WE, SEG1, SEG2, SEG3, SEG4, SEG5, SEG6, SEG7, AN1, AN2, AN3, AN4, AN5, AN6, AN7, ANIMATIE, AN, SEG);

END ARCHITECTURE;
```

12.) UCF – Fișierul de constrângeri

În fișierul de constrângeri sunt specificate locațiile pe placă pentru intrările și ieșirile folosite.

```
NET "CLK" LOC = "B8";
NET "AN(3)" LOC = "F15";
NET "AN(2)" LOC = "C18";
NET "AN(1)" LOC = "H17";
NET "AN(0)" LOC = "F17";
NET "SEG(0)" LOC = "L18";
NET "SEG(1)" LOC = "F18";
NET "SEG(2)" LOC = "D17";
NET "SEG(3)" LOC = "D16";
NET "SEG(4)" LOC = "G14";
NET "SEG(5)" LOC = "J17";
NET "SEG(6)" LOC = "H14";
NET "SEG(7)" LOC = "C17";
NET "ANIMATIE(0)" LOC="L13";
NET "ANIMATIE(1)" LOC="N17";
NET "ANIMATIE(2)" LOC="R17";
NET "RESET" LOC="K17";
NET "BUTON_IN(1)" LOC="H13";
NET "BUTON_IN(0)" LOC="B18";
NET "WE" LOC="G18";
NET "WE" CLOCK_DEDICATED_ROUTE = FALSE;
NET "BUTON_IN(1)" CLOCK_DEDICATED_ROUTE = FALSE;
NET "RESET" CLOCK_DEDICATED_ROUTE = FALSE;
```

Justificarea soluției alese

Pentru realizarea acestui proiect, am ales structurarea acestuia pe componente funcționale. Modulul principal reunește toate aceste componente.

Numele semnificative pentru intrări, ieșiri, alte semnale și variabile, împreună cu comentariile amănunțite ale codului, fac ca acesta să fie ușor de înțeles și lizibil.

Datorită structurării codului în componente, acesta este ușor de modificat și de îmbunătățit.

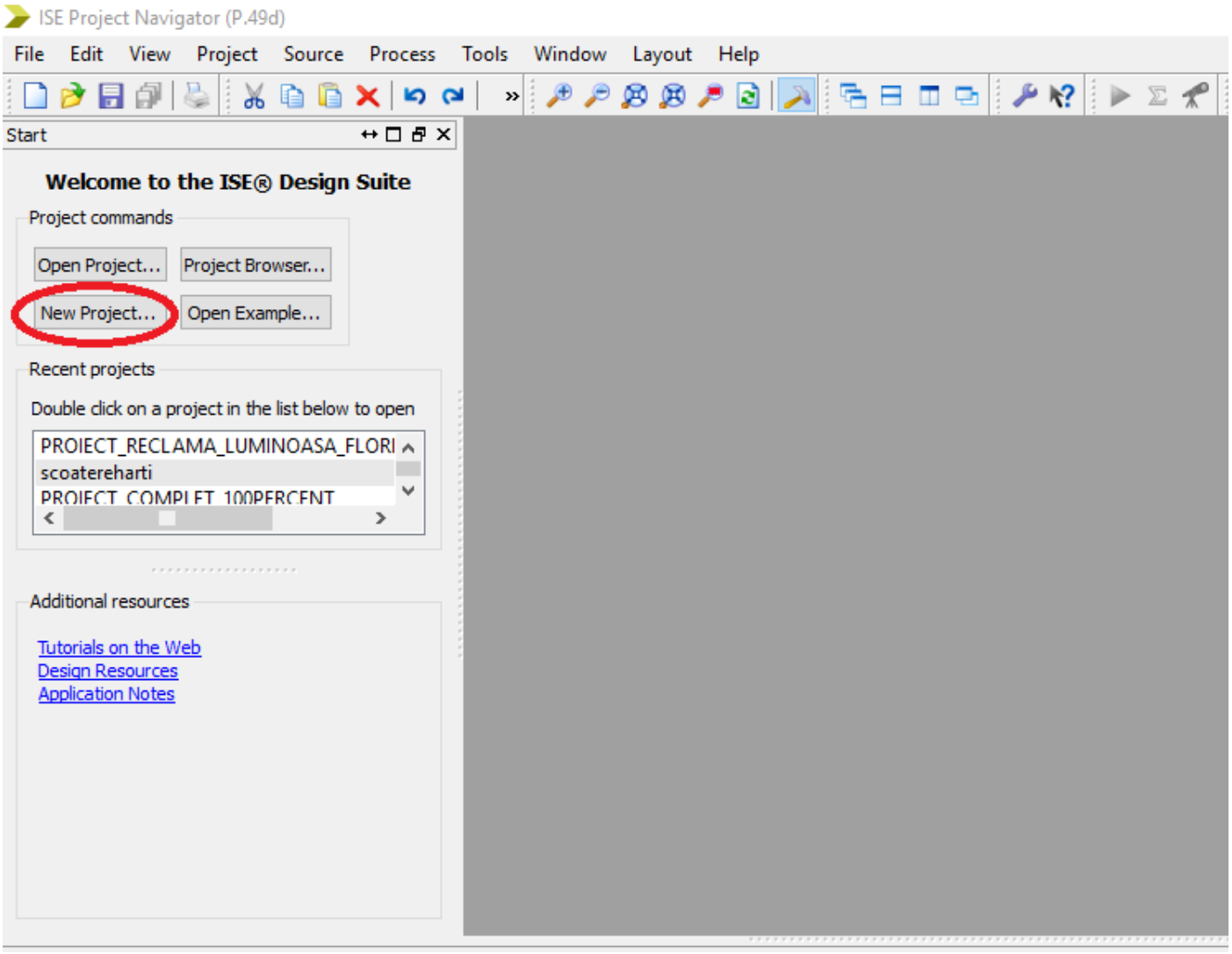
Modul în care se poate interacționa cu placa NEXYS-2 este unul intuitiv și ușor.

Instrucțiuni de utilizare

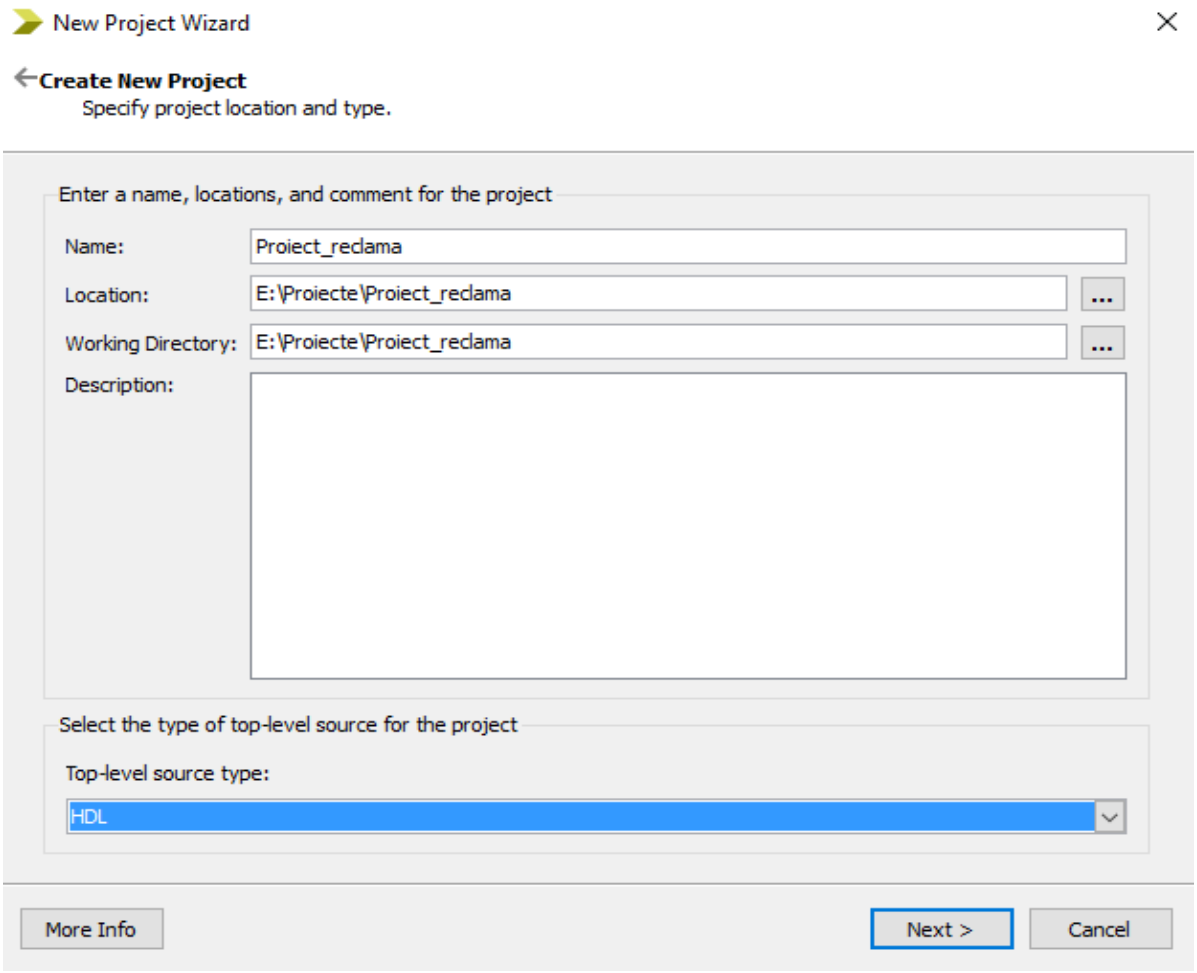
Pentru acest proiect, se utilizează placa cu FPGA NEXYS-2 a firmei DIGILENT împreună cu programul Xilinx ISE Design Suite.

Pentru utilizare se procedează după cum urmează:

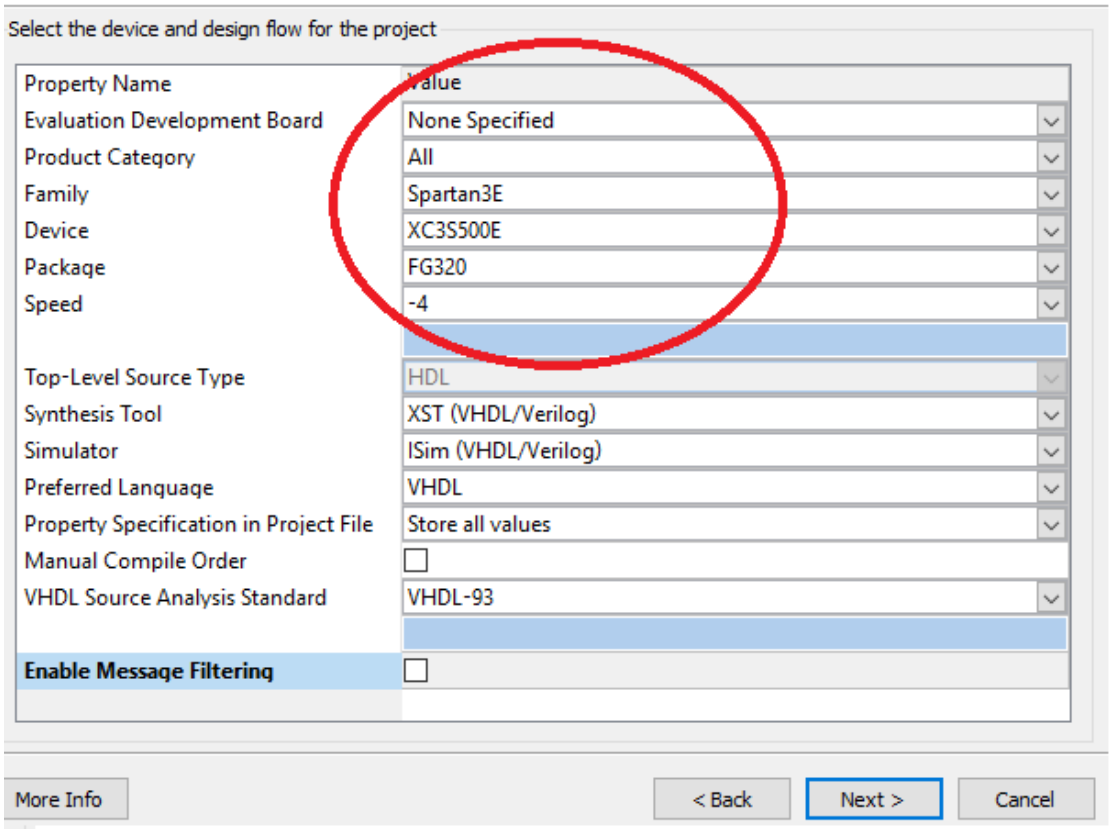
1.) Se lansează aplicația Xilinx ISE Design Suite și se creează un proiect nou.



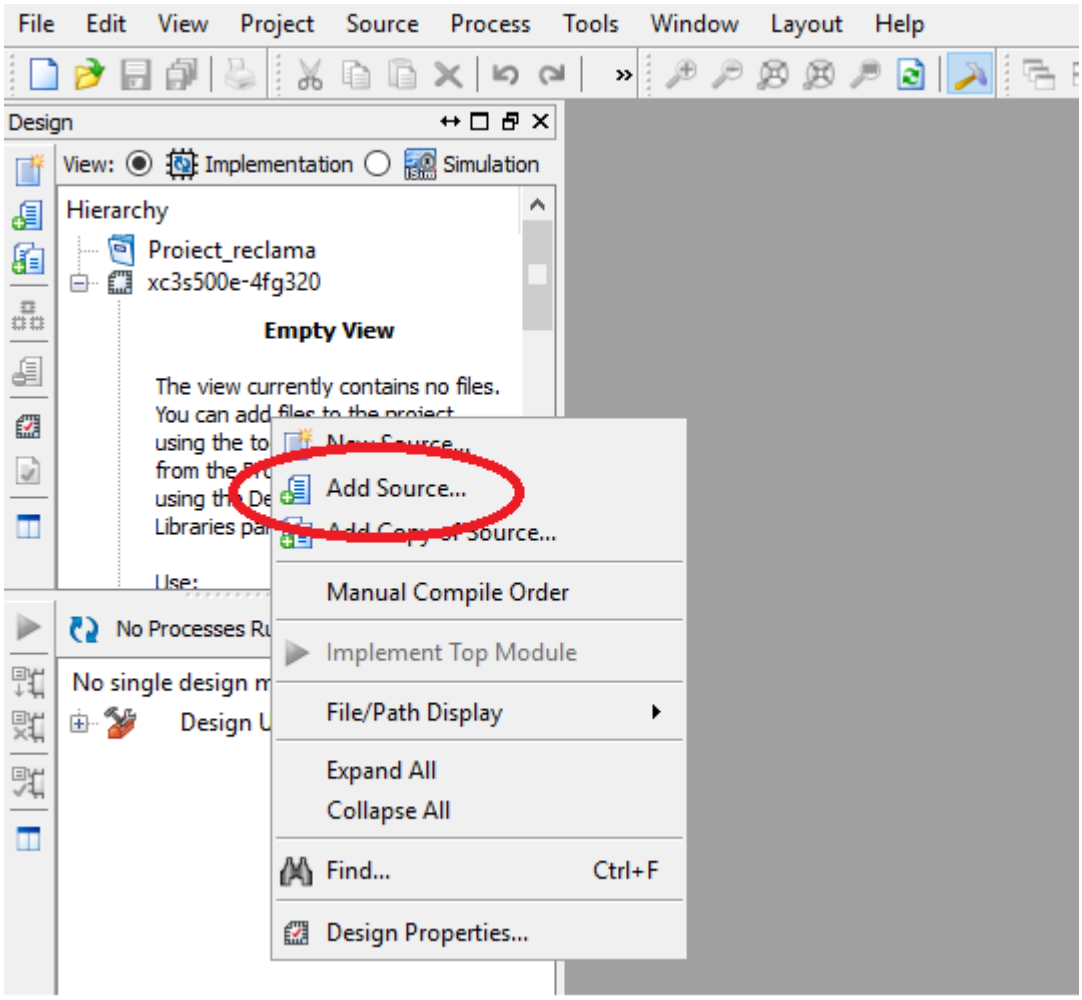
2.) Se denumește proiectul și se stabilește locul unde se salvează.



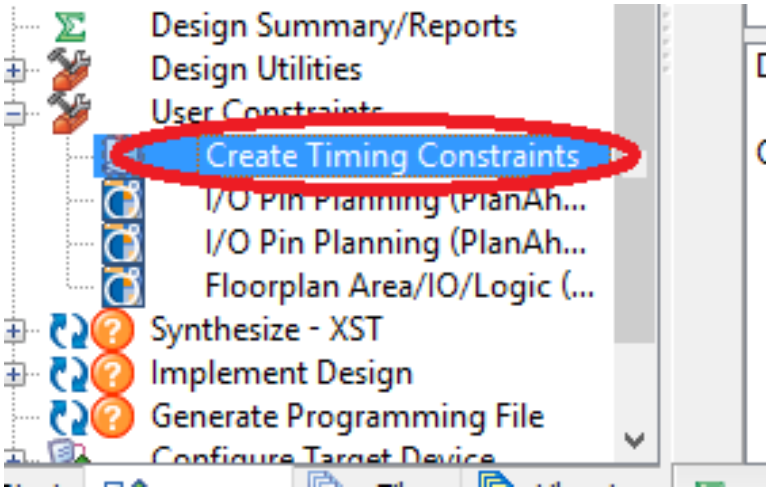
3.) Se realizează setările pentru placa NEXYS-2.



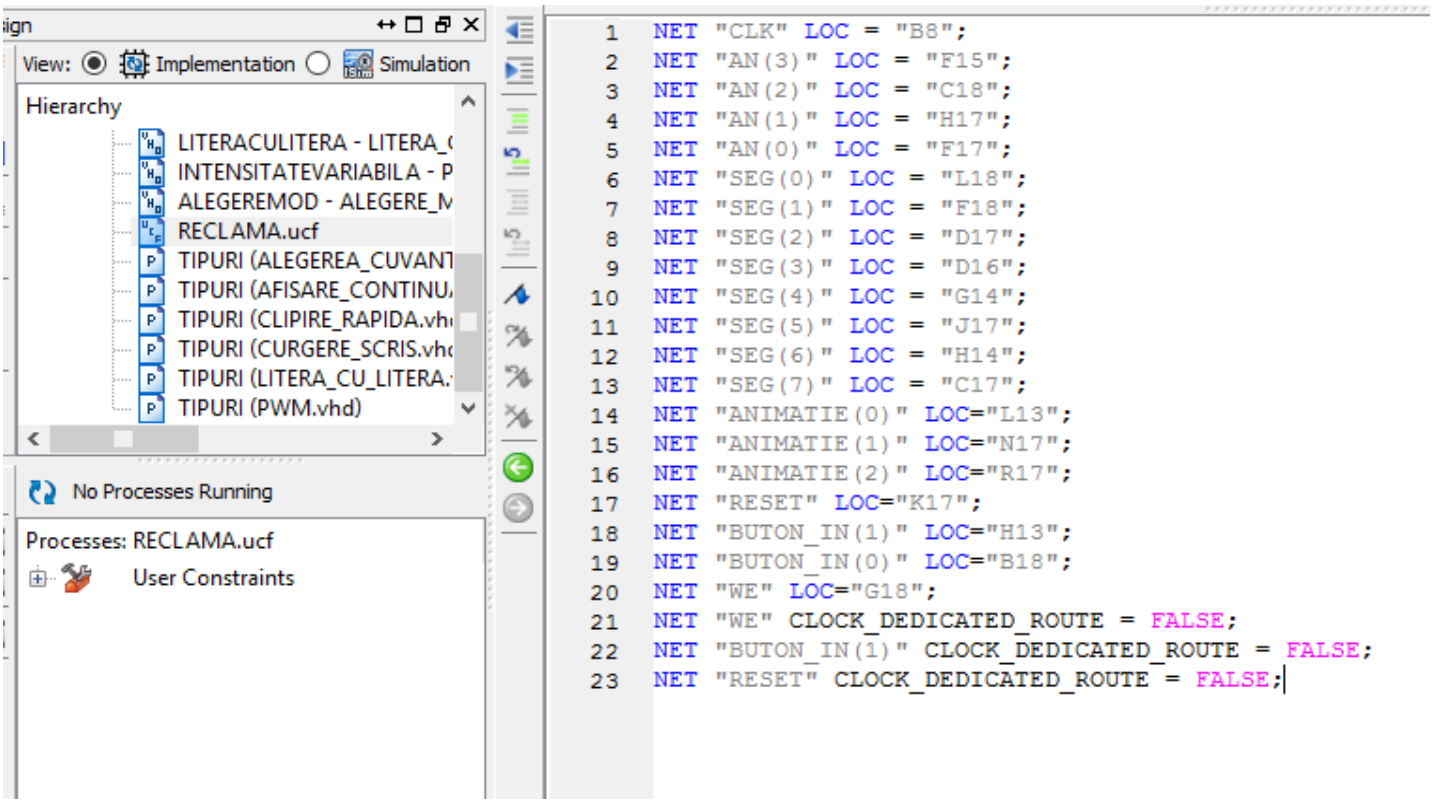
4.) Se adaugă componentele.



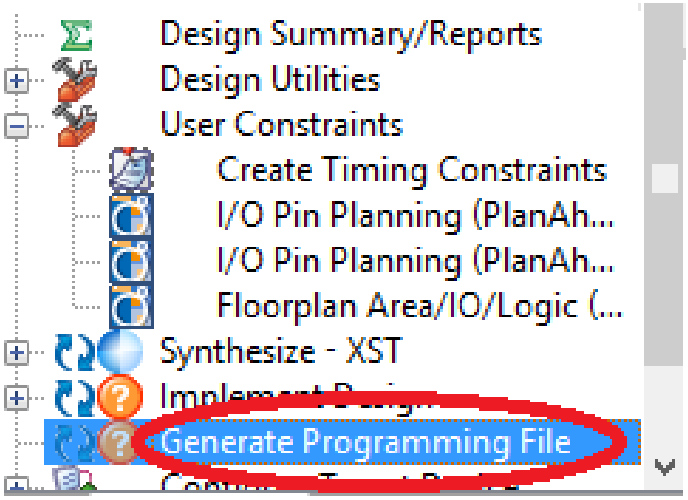
5.) Se apasă pe Create Timing Constraints, pentru a genera fișierul cu extensia ”.ucf”.



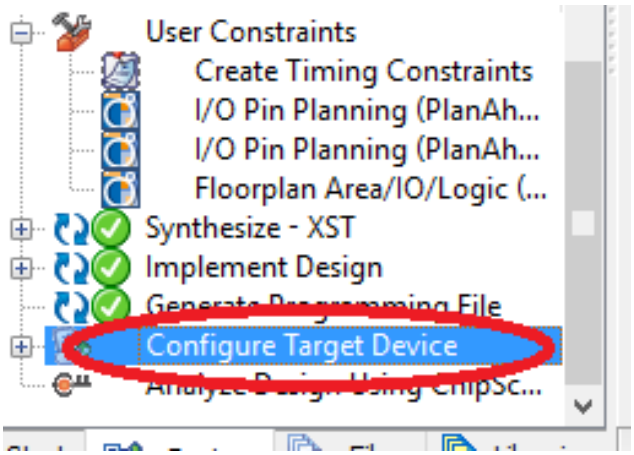
6.) Se scrie fișierul de constrângeri.



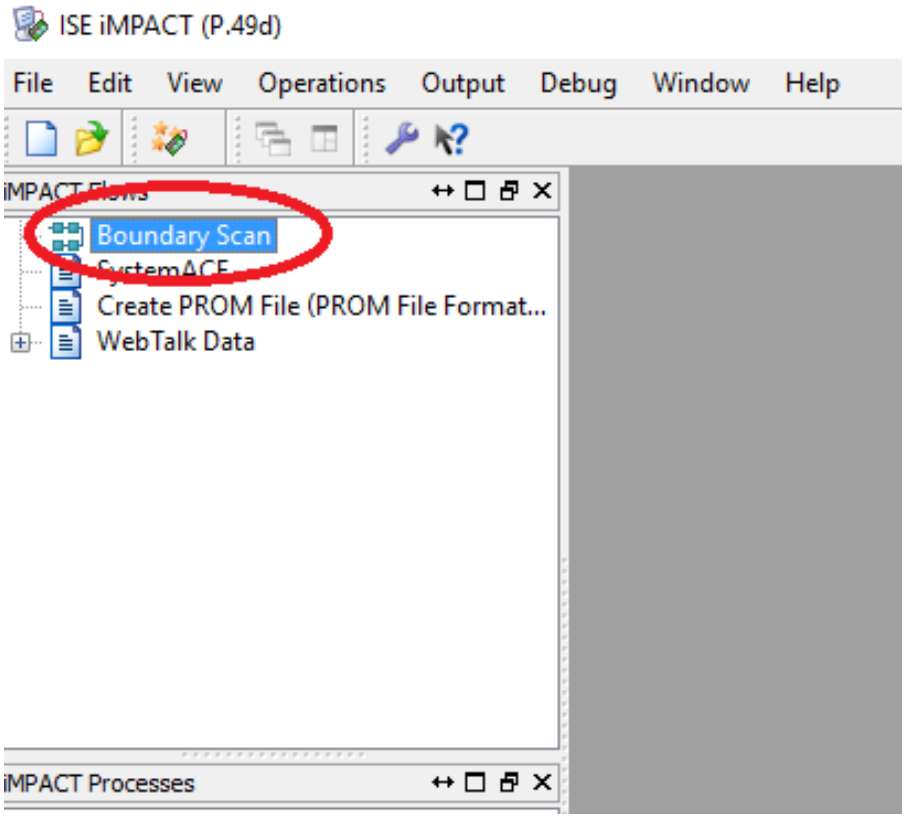
7.) Se apasă pe Generate Programming File.



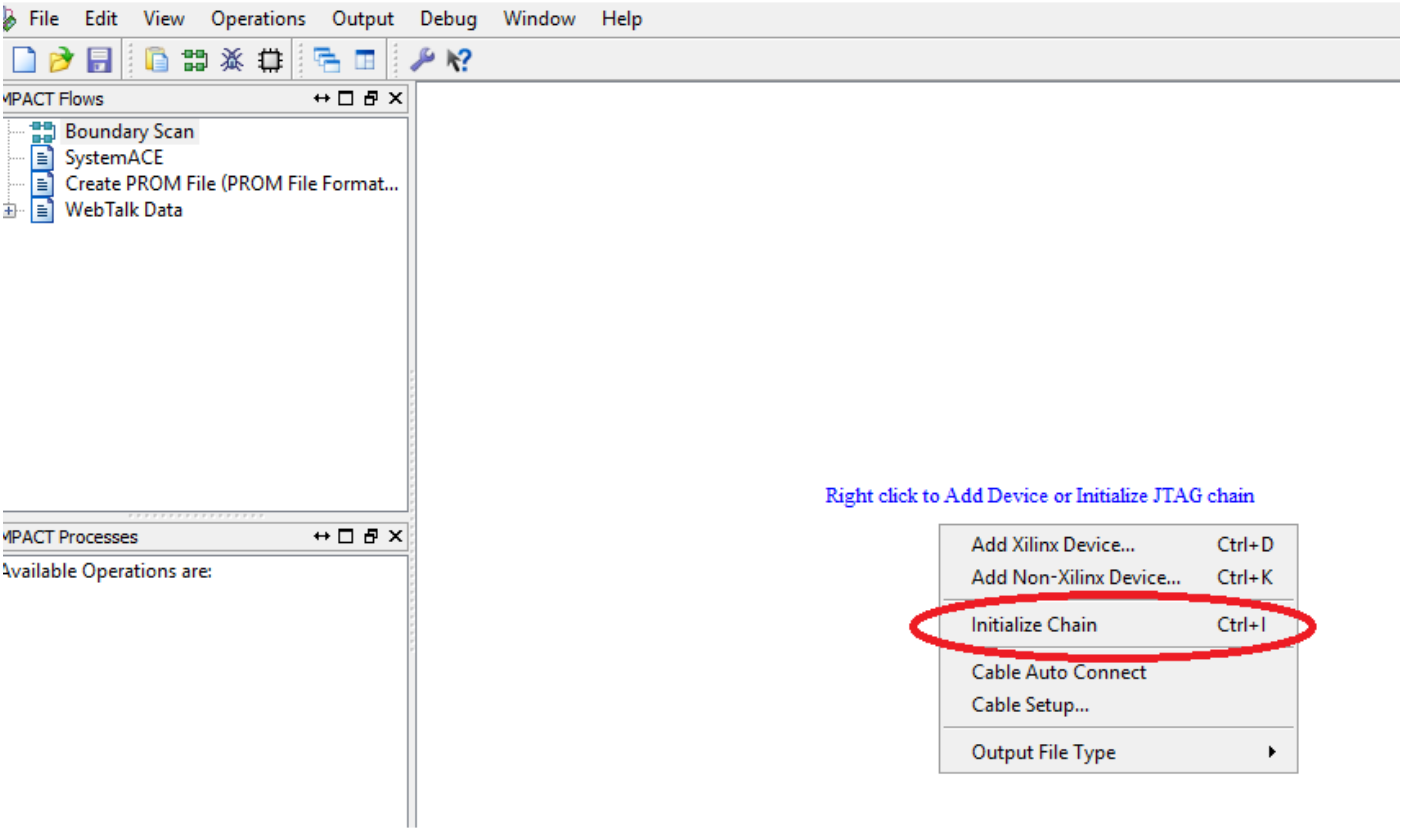
8.) Se apasă pe Configure Target Device.



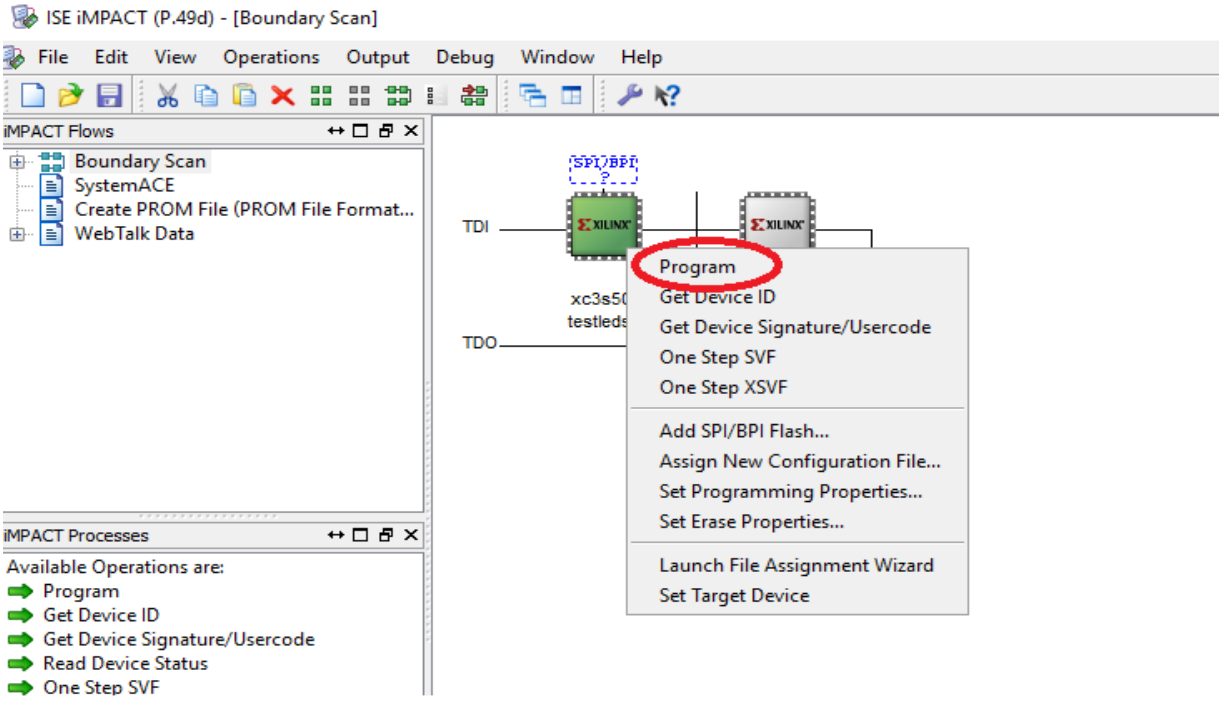
9.) După pornirea utilitarului ISE IMPACT, se apasă pe Boundary Scan.



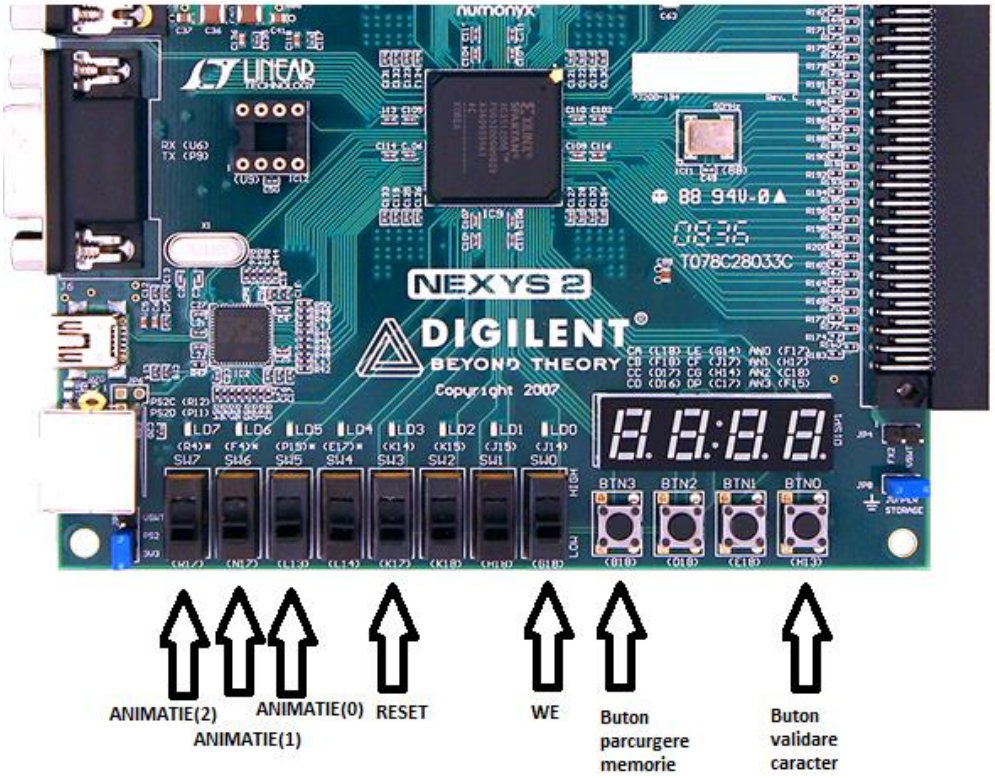
10.) Se dă click dreapta și se apasă pe Initialize Chain.



11.) Se caută fișierul ”.bit” generat de Xilinx si se apasă pe Program.



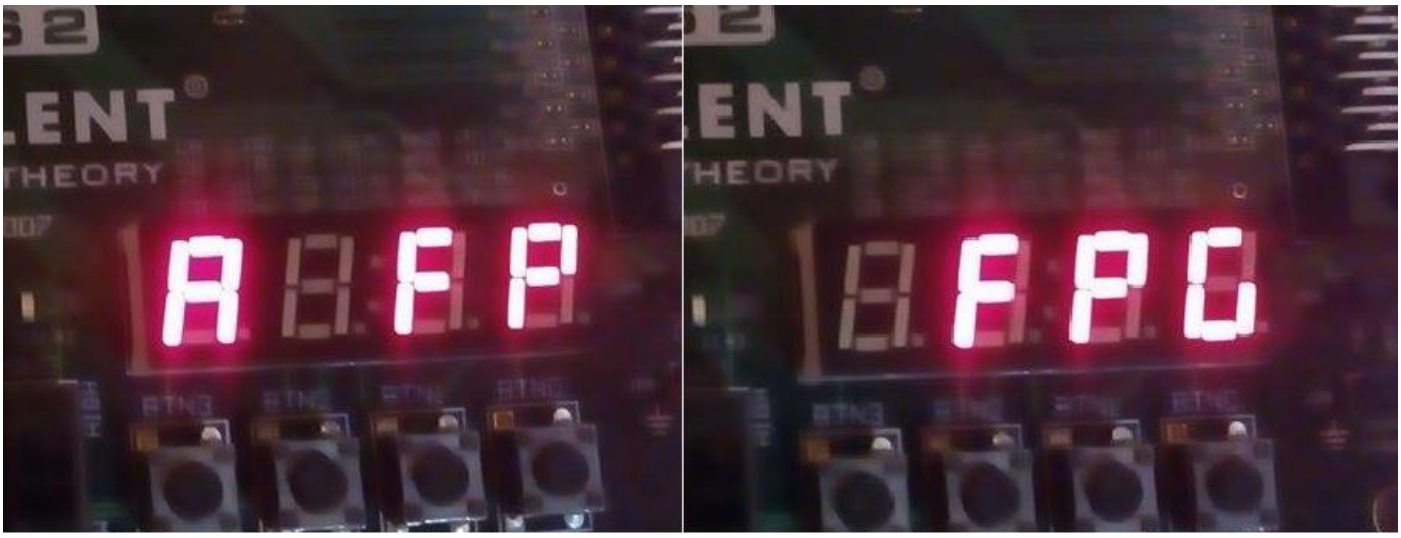
Poze cu funcționarea



Pozele cu funcționarea sunt doar pentru modurile care se pretează a fi fotografiate (afișare continuă, curgere scris, afișare literă cu literă).

1.) Afișare continuă.

2.) Curgere scris.



3.) Afișare literă cu literă.

Posibilități de dezvoltare

Printre multiplele posibilități de dezvoltare a proiectului se numără:

Posibilitatea introducerii unei reclame cu mai mult de 4 litere și a afișării acesteia în toate modurile dorite.

Utilizarea mai multor afișoare cu 7 segmente decât dispune placa cu FPGA NEXYS-2.

Utilizarea unor afișoare cu 28 de segmente în locul celor cu 7 segmente, pentru a putea reprezenta toate caracterele dorite.

Utilizarea intrării de tastatură a placii NEXYS-2 pentru introducerea reclamei, in locul celor 2 butoane.

Adăugarea mai multor moduri de afișare ale reclamei (de sus în jos, de jos în sus, curgerea scrisului din ambele părți concomitent).

Afișarea animațiilor într-o anumită ordine, fără vreo intervenție exterioară.

Această reclamă poate fi folosită ca reclamă a unui magazin, pentru a atrage privirile oamenilor.