Animație text

-FPGA Nexys3-

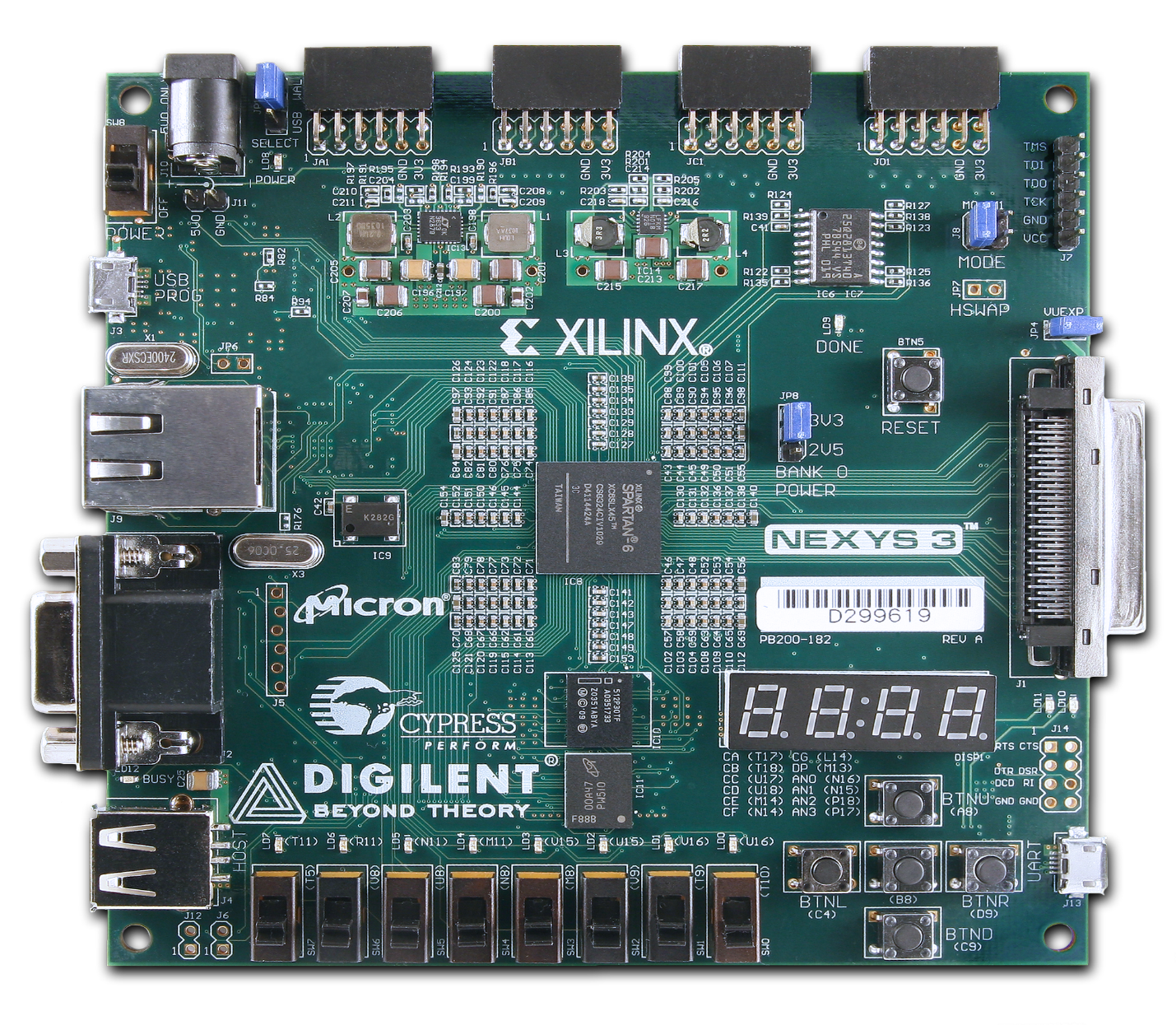
*Proiect realizat de Filip Vlad-Ilie*

*Facultatea de automatică și calculatoare*

*Anul I de studiu*

*Grupa 30211*

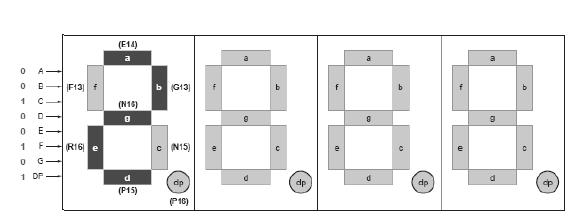
**PLACA NEXYS-3**

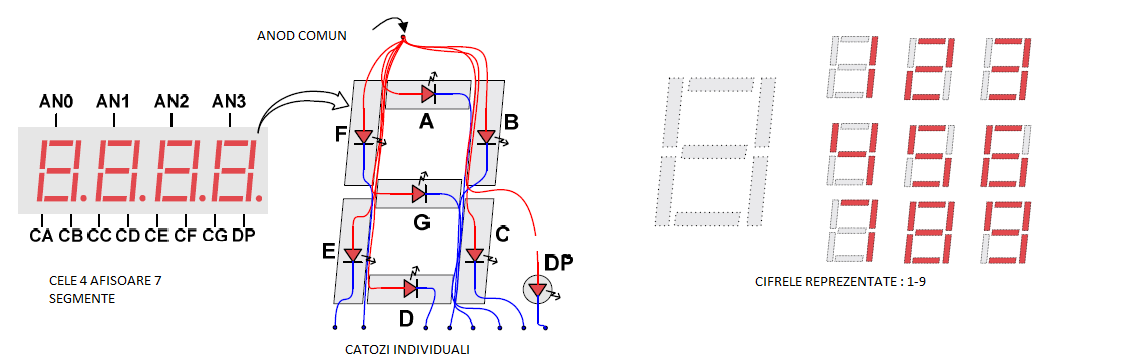
****

Fiind produsă de compania Digilent Inc., placa ***NEXYS-3*** are în principiu componente principale similare cu cele ale plăcii Spartan6.

Starter Kit: **cele 4 afișoare 7 segmente**, **cele 4 comutătoare de tip push-button** și **LED-urile.**

**a)cele 4 afișoare 7 segmente**





-după cum se observă din imagine, afișoarele au cele 7 leduri reprezentând segmentele, si un simbol **dp** reprezentând punctul decimal

-fiecare are un anod activ pe 0 (funcționează numai atunci când sunt setate pe 0 logic)

**Conexiunile dintre dispozitivul FPGA si afișajul cu 7 segmente (active pe 0)**

**Segmentul Pinul dispozitivului FPGA**

A T17

B T18

C U17

D U18

E M14

F N14

G L14

DP M13

Semnalele de control al anodului (active pe 0)

**Anode Control** AN3 AN2 AN1 AN0

**FPGA Pin** N16 N15 P18 P17

**b) Cele patru comutatoare de tip „push button”**

**Conexiunile comutatoarelor de tip „push button” la pinii dispozitivului FPGA**

**Push Button** BTN4BTN3 BTN2 BTN1 BTN0

**Pin FPGA** D9C9 C4 A8 B8

**c) LED-urile**

**Conexiunile LED-urilor la pinii dispozitivului FPGA Spartan6**

**LED** LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0

**Pin FPGA** T11 R11 N11 M11 V15 U15 V16 U16

***Clock-ul integrat în placa FPGA***

Placa NEXYS-2 include un oscilator de quartz interior de **160MHz** și o cutie pentru un al doilea oscilator .Semnalele de tact de la oscilatorii se conectează la pinii de tact global. Astfel se poate realiza divizarea de frecvență precum și multiplicarea acestuia pentru semnale de tact de frecvență variabilă.

**Componentele logice folosite pentru realizarea proiectului**

**Multiplexorul 4:1**

Cu acest multiplexor vom putea alege animația dorită din cele patru.Utilizatorul va putea alege acesta practic prin folosirea a două comutatoare de două poziții din cele 8 disponibile (mai precis primul și al doilea):



**Animația1**: comutatoarele fiind: 00 -Afisare cuvinte

**Animația2**: comutatoarele fiind: 01 - Clipire

**Animația3**: comutatoarele fiind: 10 – litera cu litera

**Animația4**: comutatoarele fiind: 11 – stanga dreapta

**Memoriile ROM**

Textul ce va trebui să fie afișat este hard-codat in codul **vhdl**. Folosim memoria ROM pentru a reprezenta literele **alfabetului englez** de la **-a- la -u-,**deoarece nu folosim toate litere pentru ca nu se pot reprezenta concret pe placa. Reprezentarea se face ținând cont de faptul că **ledurile** segmentelor sunt **active pe 0 logic**. Utilizatorul va putea alege textul dorit făcând modificări in codul **vhdl** care va fi prezentat la sfârșitul documentației.

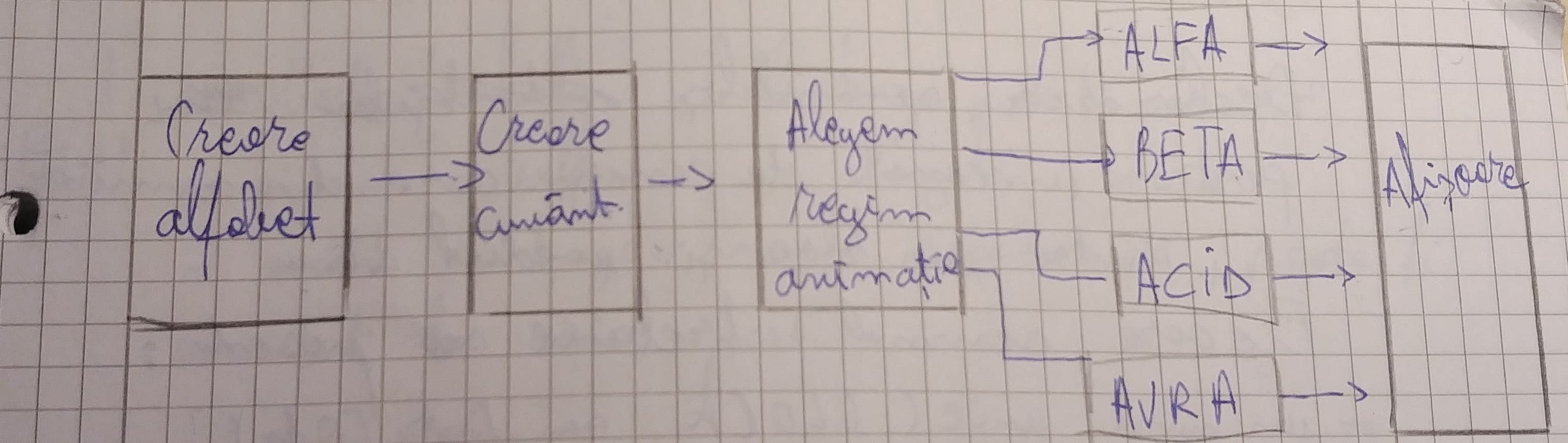
**Numărătorul pe 26 de biți**

Acesta este cel mai important element al animației fără de care utilizatorul n-ar avea posibilitatea de a vedea animația respectivă. Clock-ul plăcii **NEXYS-3** funcționează pe **160MHz**, de aceea avem nevoie de o modalitate de a incetini timpul de execuție. Folosind un numărător care va număra o secundă (sau dacă e cazul mai mult sau mai puțin) putem vedea cu ochiul liber schimbările efectuate pe afișoarele folosite.

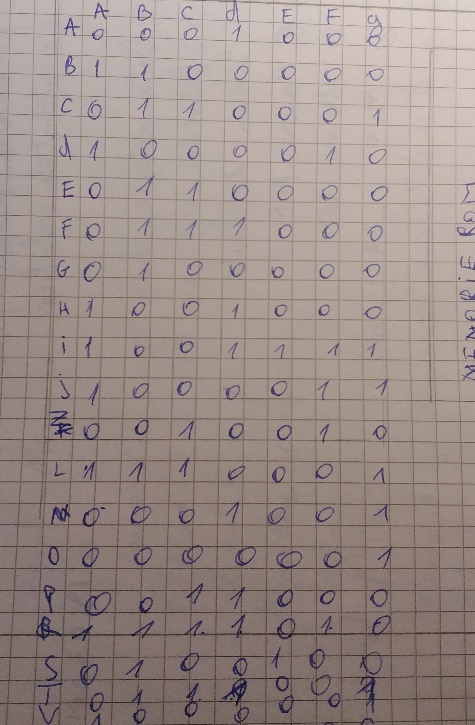
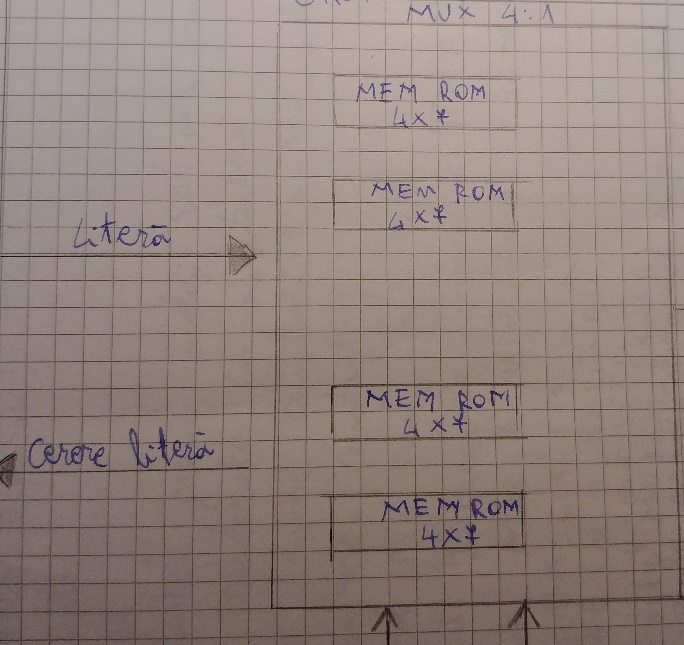
**1.SPECIFICATIA PROIECTULUI:**

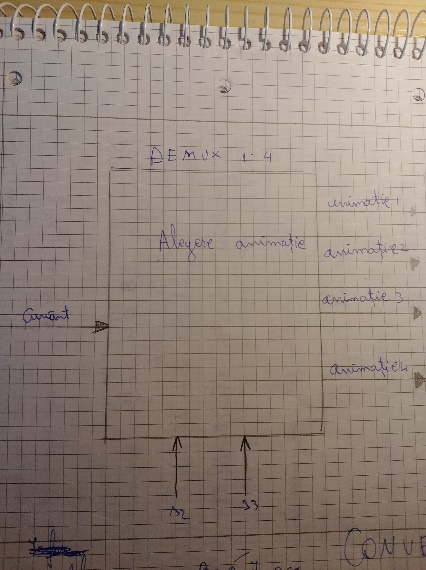
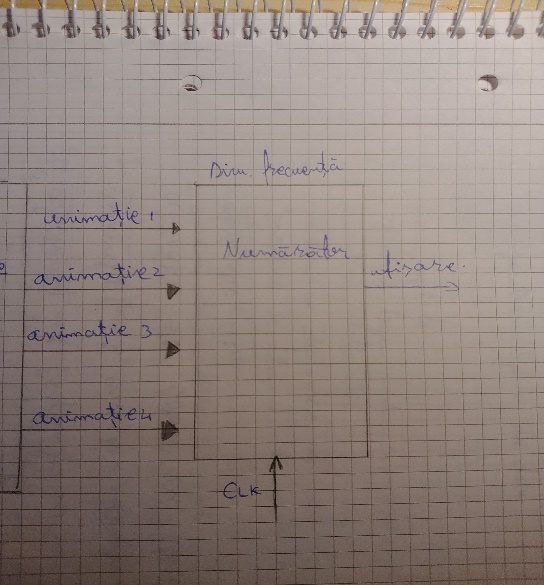
Sa se proiecteze o reclama publicitara cu animatii multiple. Se vor folosi afisajele cu 7 segmente. Textul de afisat va fi format din simboluri ale unui alfabet disponibil. Reclama va avea mai multe regimuri de functionare ce vor putea fi selectate de catre utilizator de la comutatoarele de pe placuta cu FPGA. Se va folosi oscilatorul de cuart incorporate in placuta cu FPGA. Exemple de regimuri de functionare : “curgerea” scrisului de la dreapta la stanga, palpaire, afisas litera cu litera.

**2.DESEN SCHEMA BLOC**

****

**2.1.DESEN SCHEMA BLOC DETALIATA**

**3.CODUL**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_UNSIGNED.all;

use IEEE.numeric\_std.all;

ENTITY ROM\_CUVINTE IS

PORT (

SEL : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

reset : in STD\_LOGIC;

CLK : IN STD\_LOGIC;

ANOD : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0);

CATOD : OUT STD\_LOGIC\_VECTOR (6 DOWNTO 0)

);

END ROM\_CUVINTE;

ARCHITECTURE BEH OF ROM\_CUVINTE IS

COMPONENT COUNTER is

Port ( CLK : in STD\_LOGIC;

LED1 : out STD\_LOGIC;

LED2 : out STD\_LOGIC;

LED3 : out STD\_LOGIC;

LED4 : out STD\_LOGIC);

end COMPONENT;

COMPONENT COUNTER2 is

Port ( CLK : in STD\_LOGIC;

LED1 : out STD\_LOGIC);

end COMPONENT;

TYPE LETTERS\_ALPH IS ARRAY (17 DOWNTO 0) of STD\_LOGIC\_VECTOR(6 DOWNTO 0);

type WORD1 is array (3 downto 0) of std\_logic\_vector(6 downto 0);

type WORD is array (15 downto 0) of std\_logic\_vector(6 downto 0);

type WORD22 is array (15 downto 0) of std\_logic\_vector(6 downto 0);

type WORD33 is array (15 downto 0) of std\_logic\_vector(6 downto 0);

type WORD44 is array (15 downto 0) of std\_logic\_vector(6 downto 0);

type WORD2 is array (3 downto 0) of std\_logic\_vector(6 downto 0);

type WORD3 is array (3 downto 0) of std\_logic\_vector(6 downto 0);

type WORD4 is array (3 downto 0) of std\_logic\_vector(6 downto 0);

signal LED\_activating\_counter: std\_logic\_vector(1 downto 0);

signal LED\_activating\_counter2: std\_logic\_vector(1 downto 0);

signal SELECTOR: std\_logic\_vector(1 downto 0);

SIGNAL LED\_OUT :STD\_LOGIC;

SIGNAL LED\_OUT1 :STD\_LOGIC;

SIGNAL LED\_OUT2 :STD\_LOGIC;

SIGNAL LED\_OUT3 :STD\_LOGIC;

SIGNAL LED\_OUT4 :STD\_LOGIC;

signal refresh\_counter: STD\_LOGIC\_VECTOR (19 downto 0);

signal refresh\_counter2: STD\_LOGIC\_VECTOR (26 downto 0);

CONSTANT LETTER : LETTERS\_ALPH := (0 => "1111111", --spatiu

1 => "0001000", --A

2 => "1100000", --b

3 => "0110001", --C

4 => "1000010", --d

5 => "0110000", --E

6 => "0111000", --F

7 => "0100000", --G

8 => "1001000", --H

9 => "1001111", --I

10 => "1001111", --J

11 => "1110001", --L

12 => "0001001", --n

13 => "0000001", --O

14 => "0011000", --P

15 => "1111010", --r

16 => "0100100", --S

17 => "1000001"); --U

constant W1 : WORD1 :=

(0 => letter(1), --A

1 => letter(11), --L

2 => letter(6), --F

3 => letter(1)); --A

constant W11 : WORD :=

(

0 => letter(1), --A

1 => letter(11), --L

2 => letter(6), --F

3 => letter(1), --A

4 => letter(1), --A

5 => letter(1), --A

6 => letter(11), --L

7 => letter(6), --F

8 => letter(6), --F

9 => letter(1), --A

10 => letter(1), --A

11 => letter(11), --L

12 => letter(6), --F

13 => letter(11), --L

14 => letter(1), --A

15 => letter(1)); --A

constant W2 : WORD2 := ( 0 => letter(3),--C

1 => letter(13),--O

2 => letter(11),--L

3 => letter(1));--A

constant W22 : WORD22 := (

0 => letter(3),--C

1 => letter(13),--O

2 => letter(11),--L

3 => letter(1),--A

4 => letter(1),--A

5 => letter(3),--C

6 => letter(13),--O

7 => letter(11),--L

8 => letter(11),--L

9 => letter(1),--A

10 => letter(3),--C

11 => letter(13),--O

12 => letter(13),--O

13 => letter(11),--L

14 => letter(1),--A

15 => letter(3));--C

constant W3: WORD3 :=

(0 => letter(2),--B

1=> letter(5), --E

2=> letter(3), --C

3=> letter(9)); --I

constant W33: WORD33 :=

(0 => letter(2),--B

1=> letter(5), --E

2=> letter(3), --C

3=> letter(9), --I

4 => letter(9),--I

5=> letter(2), --B

6=> letter(5), --E

7=> letter(3), --C

8 => letter(3),--C

9 => letter(9), --I

10=> letter(2), --B

11 => letter(5), --E

12 => letter(5),--E

13 => letter(3), --C

14 => letter(9), --I

15 => letter(2)); --B

constant W4: WORD4 := (0 => letter(14), --P

1 => letter(13),--0

2 => letter(15), --R

3 => letter(3));--C

constant W44: WORD44 := (

0 => letter(14), --P

1 => letter(13),--0

2 => letter(15), --R

3 => letter(3),--C

4 => letter(3), --C

5 => letter(14),--P

6 => letter(13), --O

7 => letter(15), --R

8 => letter(15), --R

9 => letter(3),--C

10 => letter(14), --P

11 => letter(13),--0

12 => letter(13), --0

13 => letter(15),--R

14 => letter(3), --C

15 => letter(14));--P

begin

LEDURI: COUNTER PORT MAP(

CLK => CLK,

LED1=>LED\_OUT,

LED2=>LED\_OUT2,

LED3=>LED\_OUT3,

LED4=>LED\_OUT4);

LEDURI2: COUNTER2 PORT MAP(

CLK => CLK,

LED1=>LED\_OUT1);

clk1 : process(clk,RESET)

begin

if(reset='1') then

refresh\_counter <= (others => '0');

refresh\_counter2 <=(others =>'0');

elsif(rising\_edge(clk)) then

refresh\_counter <= refresh\_counter + 1;

refresh\_counter2 <= refresh\_counter2 +1;

end if;

end process;

LED\_activating\_counter <= refresh\_counter (19 downto 18);

LED\_activating\_counter2 <= refresh\_counter2 (26 downto 25);

LED : process(LED\_activating\_counter,sel,LED\_OUT,LED\_OUT2,LED\_OUT3,LED\_OUT4,LED\_OUT1,LED\_ACTIVATING\_COUNTER2)

variable i:integer:=0;

variable k:integer:=0;

begin

case sel is

-------------------------------------------------------------- PRIMUL CUVANT TOATE ANIMMATIILE

when "0000" =>

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W1(0);

when "01" =>

ANOD <="1011";

CATOD <= W1(1);

when "10" =>

ANOD <="1101";

CATOD <= W1(2);

when others =>

ANOD <="1110";

CATOD <= W1(3);

end case;

when "0001" =>

if LED\_OUT1='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W1(0);

when "01" =>

ANOD <="1011";

CATOD <= W1(1);

when "10" =>

ANOD <="1101";

CATOD <= W1(2);

when others =>

ANOD <="1110";

CATOD <= W1(3);

end case;

else

ANOD <="1111";

CATOD <="1111111";

end if;

when "0010" =>

case LED\_ACTIVATING\_COUNTER2 is

when "00" =>

ANOD <="0111";

CATOD <= W1(0);

when "01" =>

ANOD <="1011";

CATOD <= W1(1);

when "10" =>

ANOD <="1101";

CATOD <= W1(2);

when others =>

ANOD <="1110";

CATOD <= W1(3);

end case;

WHEN "0011" =>

if led\_out ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W11(0);

when "01" =>

ANOD <="1011";

CATOD <= W11(1);

when "10" =>

ANOD <="1101";

CATOD <= W11(2);

when others =>

ANOD <="1110";

CATOD <= W11(3);

end case;

else

IF LED\_OUT2 ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W11(4);

when "01" =>

ANOD <="1011";

CATOD <= W11(5);

when "10" =>

ANOD <="1101";

CATOD <= W11(6);

when others =>

ANOD <="1110";

CATOD <= W11(7);

end case;

ELSE

IF LED\_OUT3 ='1' THEN

Case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W11(8);

when "01" =>

ANOD <="1011";

CATOD <= W11(9);

when "10" =>

ANOD <="1101";

CATOD <= W11(10);

when others =>

ANOD <="1110";

CATOD <= W11(11);

end case;

ELSE

IF LED\_OUT4='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W11(12);

when "01" =>

ANOD <="1011";

CATOD <= W11(13);

when "10" =>

ANOD <="1101";

CATOD <= W11(14);

when others =>

ANOD <="1110";

CATOD <= W11(15);

end case;

else

ANOD <="1111";

CATOD <="1111111";

END IF;

END IF;

END IF;

END IF;

-------------------------------------------------------------- PRIMUL CUVANT TOATE ANIMMATIILE

-------------------------------------------------------------- AL DOILEA CUVANT TOATE ANIMMATIILE

when "0100" =>

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W2(0);

when "01" =>

ANOD <="1011";

CATOD <= W2(1);

when "10" =>

ANOD <="1101";

CATOD <= W2(2);

when others =>

ANOD <="1110";

CATOD <= W2(3);

end case;

when "0101" =>

if LED\_OUT1='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W2(0);

when "01" =>

ANOD <="1011";

CATOD <= W2(1);

when "10" =>

ANOD <="1101";

CATOD <= W2(2);

when others =>

ANOD <="1110";

CATOD <= W2(3);

end case;

else

ANOD <="1111";

CATOD <="1111111";

end if;

when "0110" =>

case LED\_ACTIVATING\_COUNTER2 is

when "00" =>

ANOD <="0111";

CATOD <= W2(0);

when "01" =>

ANOD <="1011";

CATOD <= W2(1);

when "10" =>

ANOD <="1101";

CATOD <= W2(2);

when others =>

ANOD <="1110";

CATOD <= W2(3);

end case;

WHEN "0111" =>

if led\_out ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W22(0);

when "01" =>

ANOD <="1011";

CATOD <= W22(1);

when "10" =>

ANOD <="1101";

CATOD <= W22(2);

when others =>

ANOD <="1110";

CATOD <= W22(3);

end case;

else

IF LED\_OUT2 ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W22(4);

when "01" =>

ANOD <="1011";

CATOD <= W22(5);

when "10" =>

ANOD <="1101";

CATOD <= W22(6);

when others =>

ANOD <="1110";

CATOD <= W22(7);

end case;

ELSE

IF LED\_OUT3 ='1' THEN

Case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W22(8);

when "01" =>

ANOD <="1011";

CATOD <= W22(9);

when "10" =>

ANOD <="1101";

CATOD <= W22(10);

when others =>

ANOD <="1110";

CATOD <= W22(11);

end case;

ELSE

IF LED\_OUT4='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W22(12);

when "01" =>

ANOD <="1011";

CATOD <= W22(13);

when "10" =>

ANOD <="1101";

CATOD <= W22(14);

when others =>

ANOD <="1110";

CATOD <= W22(15);

end case;

else

ANOD <="1111";

CATOD <="1111111";

END IF;

END IF;

END IF;

END IF;

-------------------------------------------------------------- AL DOILEA CUVANT TOATE ANIMMATIILE

-------------------------------------------------------------- AL TREILEA CUVANT TOATE ANIMMATIILE

when "1000" => case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W3(0);

when "01" =>

ANOD <="1011";

CATOD <= W3(1);

when "10" =>

ANOD <="1101";

CATOD <= W3(2);

when others =>

ANOD <="1110";

CATOD <= W3(3);

end case;

when "1001" =>

if LED\_OUT1='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W3(0);

when "01" =>

ANOD <="1011";

CATOD <= W3(1);

when "10" =>

ANOD <="1101";

CATOD <= W3(2);

when others =>

ANOD <="1110";

CATOD <= W3(3);

end case;

else

ANOD <="1111";

CATOD <="1111111";

end if;

when "1010" =>

case LED\_ACTIVATING\_COUNTER2 is

when "00" =>

ANOD <="0111";

CATOD <= W3(0);

when "01" =>

ANOD <="1011";

CATOD <= W3(1);

when "10" =>

ANOD <="1101";

CATOD <= W3(2);

when others =>

ANOD <="1110";

CATOD <= W3(3);

end case;

WHEN "1011" =>

if led\_out ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W33(0);

when "01" =>

ANOD <="1011";

CATOD <= W33(1);

when "10" =>

ANOD <="1101";

CATOD <= W33(2);

when others =>

ANOD <="1110";

CATOD <= W33(3);

end case;

else

IF LED\_OUT2 ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W33(4);

when "01" =>

ANOD <="1011";

CATOD <= W33(5);

when "10" =>

ANOD <="1101";

CATOD <= W33(6);

when others =>

ANOD <="1110";

CATOD <= W33(7);

end case;

ELSE

IF LED\_OUT3 ='1' THEN

Case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W33(8);

when "01" =>

ANOD <="1011";

CATOD <= W33(9);

when "10" =>

ANOD <="1101";

CATOD <= W33(10);

when others =>

ANOD <="1110";

CATOD <= W33(11);

end case;

ELSE

IF LED\_OUT4='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W33(12);

when "01" =>

ANOD <="1011";

CATOD <= W33(13);

when "10" =>

ANOD <="1101";

CATOD <= W33(14);

when others =>

ANOD <="1110";

CATOD <= W33(15);

end case;

else

ANOD <="1111";

CATOD <="1111111";

END IF;

END IF;

END IF;

END IF;

-------------------------------------------------------------- AL TREILEA CUVANT TOATE ANIMATIILE

-------------------------------------------------------------- AL PATRULEA CUVANT TOATE ANIMMATIILE

when "1100" =>

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W4(0);

when "01" =>

ANOD <="1011";

CATOD <= W4(1);

when "10" =>

ANOD <="1101";

CATOD <= W4(2);

when others =>

ANOD <="1110";

CATOD <= W4(3);

end case;

when "1101" =>

if LED\_OUT1='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W4(0);

when "01" =>

ANOD <="1011";

CATOD <= W4(1);

when "10" =>

ANOD <="1101";

CATOD <= W4(2);

when others =>

ANOD <="1110";

CATOD <= W4(3);

end case;

else

ANOD <="1111";

CATOD <="1111111";

end if;

when "1110" =>

case LED\_ACTIVATING\_COUNTER2 is

when "00" =>

ANOD <="0111";

CATOD <= W4(0);

when "01" =>

ANOD <="1011";

CATOD <= W4(1);

when "10" =>

ANOD <="1101";

CATOD <= W4(2);

when others =>

ANOD <="1110";

CATOD <= W4(3);

end case;

WHEN "1111" =>

if led\_out ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W44(0);

when "01" =>

ANOD <="1011";

CATOD <= W44(1);

when "10" =>

ANOD <="1101";

CATOD <= W44(2);

when others =>

ANOD <="1110";

CATOD <= W44(3);

end case;

else

IF LED\_OUT2 ='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W44(4);

when "01" =>

ANOD <="1011";

CATOD <= W44(5);

when "10" =>

ANOD <="1101";

CATOD <= W44(6);

when others =>

ANOD <="1110";

CATOD <= W44(7);

end case;

ELSE

IF LED\_OUT3 ='1' THEN

Case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W44(8);

when "01" =>

ANOD <="1011";

CATOD <= W44(9);

when "10" =>

ANOD <="1101";

CATOD <= W44(10);

when others =>

ANOD <="1110";

CATOD <= W44(11);

end case;

ELSE

IF LED\_OUT4='1' THEN

case LED\_ACTIVATING\_COUNTER is

when "00" =>

ANOD <="0111";

CATOD <= W44(12);

when "01" =>

ANOD <="1011";

CATOD <= W44(13);

when "10" =>

ANOD <="1101";

CATOD <= W44(14);

when others =>

ANOD <="1110";

CATOD <= W44(15);

end case;

else

ANOD <="1111";

CATOD <="1111111";

END IF;

END IF;

END IF;

END IF;

WHEN OTHERS =>

ANOD <="1111";

CATOD <="1111111";

END CASE;

END PROCESS;

END ARCHITECTURE ;

**4.COMPONENTE UTILIZATE**

-Divizor de frecventa

-Demux 4 la 1

-Memorie ROM

**5.SEMNIFICATIA NOTIUNILOR**

ANOD – anodurile de pe placa

CATOD- catodurile de pe placa

RESET -RESET

CLK-clock-ul de pe placa

Led\_out : represinta semnalul pentru afisarea primului cuvant-clock-ul divizat

Led\_out2 : represinta semnalul pentru afisarea al doilea cuvant-clock-ul divizat

Led\_out 3: represinta semnalul pentru afisarea al treiela cuvant-clock-ul divizat

Led\_out 4: represinta semnalul pentru afisarea al patrulea cuvant-clock-ul divizat

SEL- selectia

LITTERS\_ALPH: reprezinta alfabetul creat

WORD1: Tablou de 4x7 care reprezinta cuvantul in sine 1

WORD2: Tablou de 4x7 care reprezinta cuvantul in sine 2

WORD3: Tablou de 4x7 care reprezinta cuvantul in sine 3

WORD4: Tablou de 4x7 care reprezinta cuvantul in sine 4

WORD11: Tablou de 16x7 care reprezinta toate posibilitatile de shiftare al cuvantului 1

WORD22: Tablou de 16x7 care reprezinta toate posibilitatile de shiftare al cuvantului 2

WORD33: Tablou de 16x7 care reprezinta toate posibilitatile de shiftare al cuvantului 3

WORD44: Tablou de 16x7 care reprezinta toate posibilitatile de shiftare al cuvantului 4

LED\_activating\_counter :reprezinta semnalul pentru afisare la clipire a tuturor cuvintelor

LED\_activating\_counter2 :reprezinta semnalul pentru afisare litera cu litera

refresh\_counter: numara clokc-ul pentru afisare

refresh\_counter2: Numara pentru afisare litera cu litera

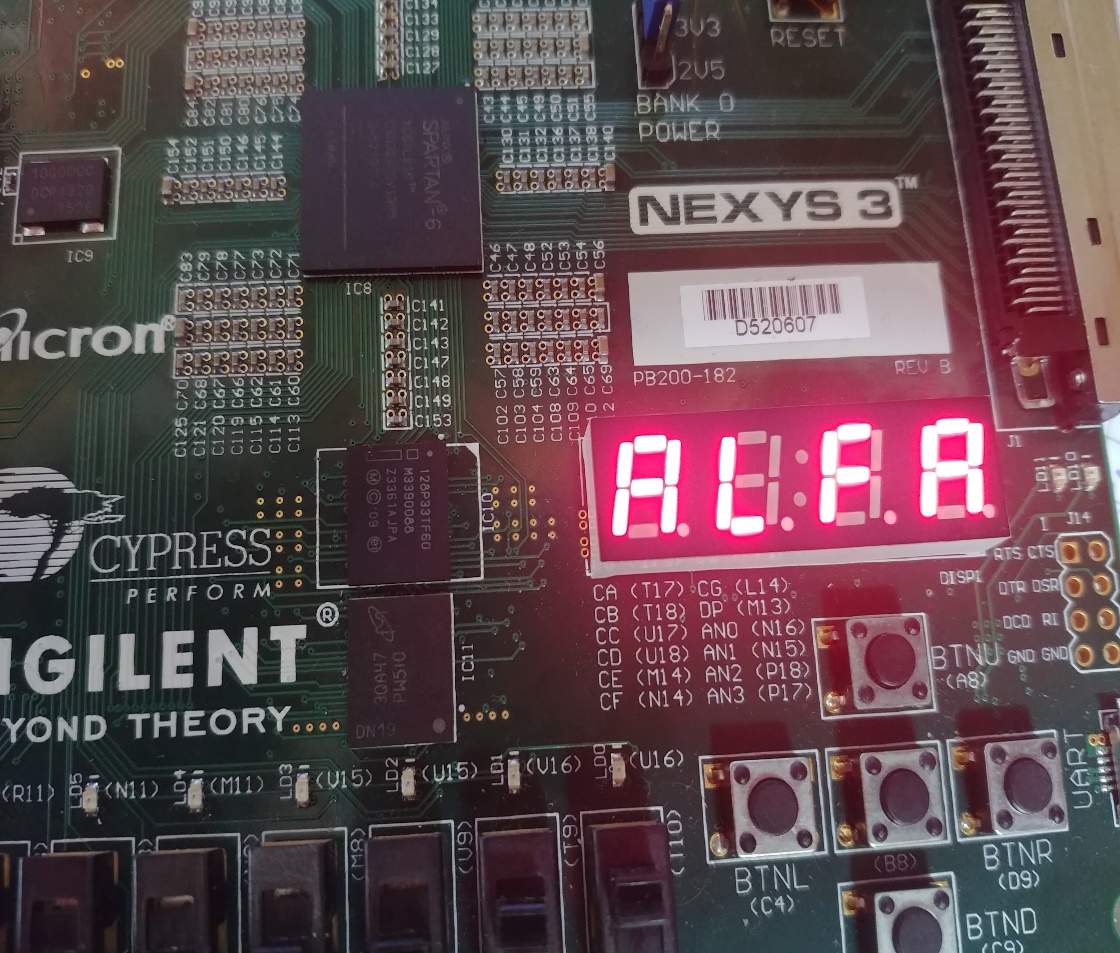
**6. JUSTIFICAREA SOLUTIEI ALESE**

Am ales aceasta solutie deoarece aplic in acest program cunostiintele invatate in acest semestru si m-a ajutat sa inteleg acest limbaj hardware. Solutia mea nu este cea mai eficienta dar rezolva cerintele cerute, desigur ca se poate updata si forma o solutie eficienta

**7. UTILIZARE SI REZULTATE**

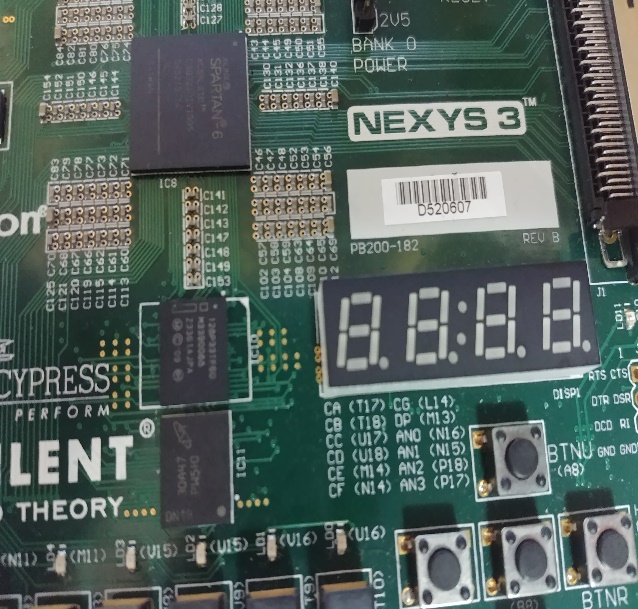
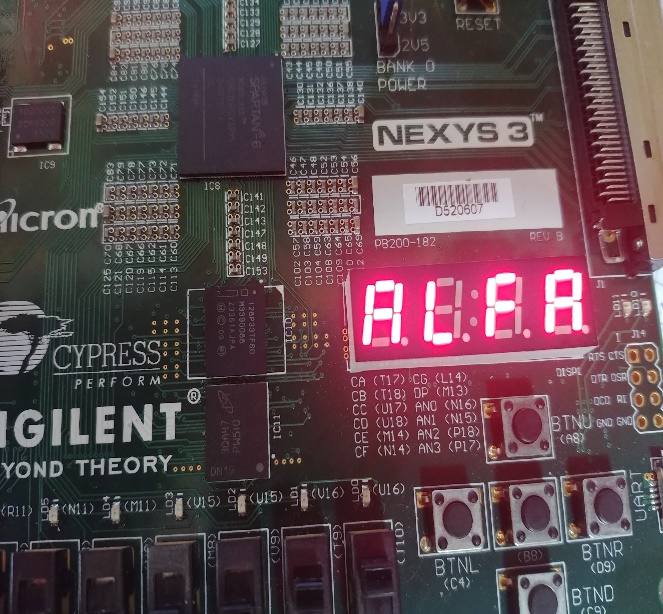
ANIMATIA 1: APARE CUVANTUL

Cu toate ca pare o animatie simpla,nu este ,totul tine de clock si de cum il modelezi pe placa, cu refresh\_ccounter numaram clock-ul si cu led\_activating afisam pe fiecare separate cu un demux foarte repede pe fiiecare incat ochiul nu face diferenta.



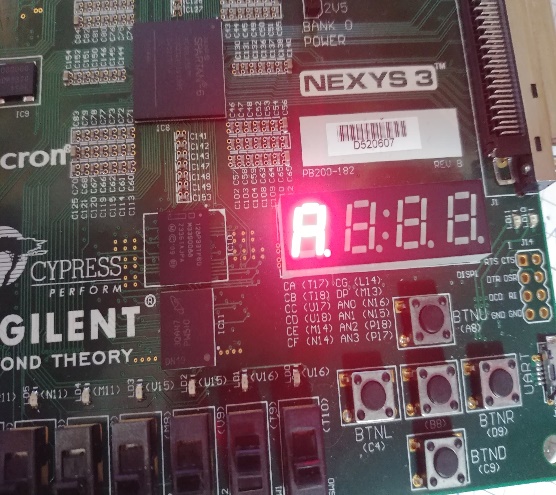
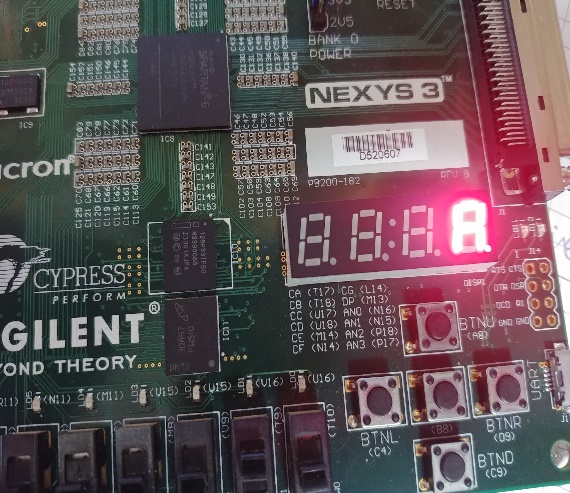
ANIMATIA 2: CLIPIRE CUVANT

Se foloseste un contor care masoara clock-ul si la final in functie de cat de mare vrei sa fie contorul apare si dispare cuvantul. Si de cate ori led\_out e 1 atunci se face aproximativ o secunda si se poate afisa cuvantul sau sa nu apara.

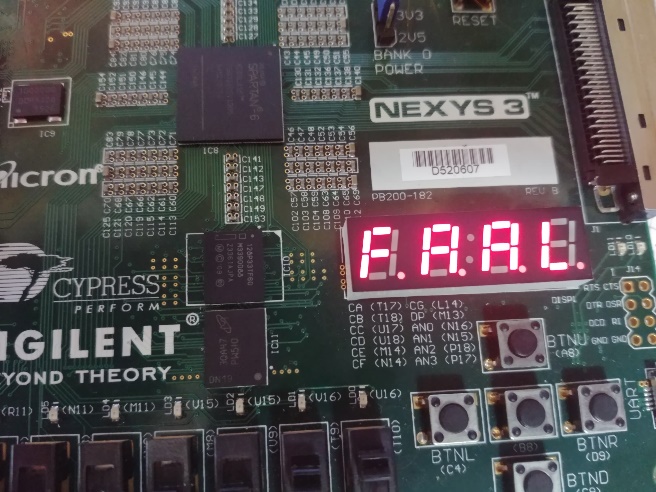
ANIMATIA 3: CLIPIRE TOATE LITERELE

De asemenea clock-ul joaca un rol foarte important deoarece tot cu ajutorul acestuia si a afisorului ajungem sa afisam litera cu litera. Refresh\_counter2 numara pana la 26 de biti astfel cand se afiseaza cuvantul se face litera cu litera.

ANIMATIA 4: STANGA->DREAPTA

Se folosesc mai multi contori care Numara clock-ul si in functie de fiecare se afiseaza cate un cuvant din cele 4.

**8.POSIBILITATI DE DEZVOLTTARE ULTERIOARA**

- Ulterior se pot folosi registre pentru a face shiftarile de la stanga la dreapta

- Se pot folosi mai putine if-uri sau case-uri pentru eficienta