MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Proiect VHDL: Reclama publicitara cu animatii multiple

-FPGA Basys 3 -

Autor: Dediu Alexandra-Petruta
Profesor coordonator: Daian Lorena

Cuprins

1 Introducere

1.1 Informatii de baza despre Basys 3

Placa Basys3 este o platformă de dezvoltare digitală completă, gata de utilizat, bazată pe cea mai recentă arhitectură de porți programabilă Fieldbus Programmable (FPGA) Artix-7 ™ de la Xilinx.

FPGA Artix-7 este optimizata pentru o logică de înaltă performanță și oferă o capacitate mai mare, performanță mai mare și mai multe resurse decât modelele anterioare. Include:

• 33.280 de celule logice în 5.200 de slices

• 1.800 Kbits de memorie RAM rapidă;

• Viteza clockului intern depaseste 450MHz;

Basys3 are de asemenea, următoarele caracteristici I / O: 16 comutatoare, 5 butoane, 4 segmente de afișare LED pe 7 segmente, ieșire VGA de 12 pini. De asemenea, are un port USB-JTAG pentru programare și comunicare FPGA.

Programarea acestei plăci se face în VIVADO Design Suite (oferită de XILINX) utilizând VHDL sau Verilog.

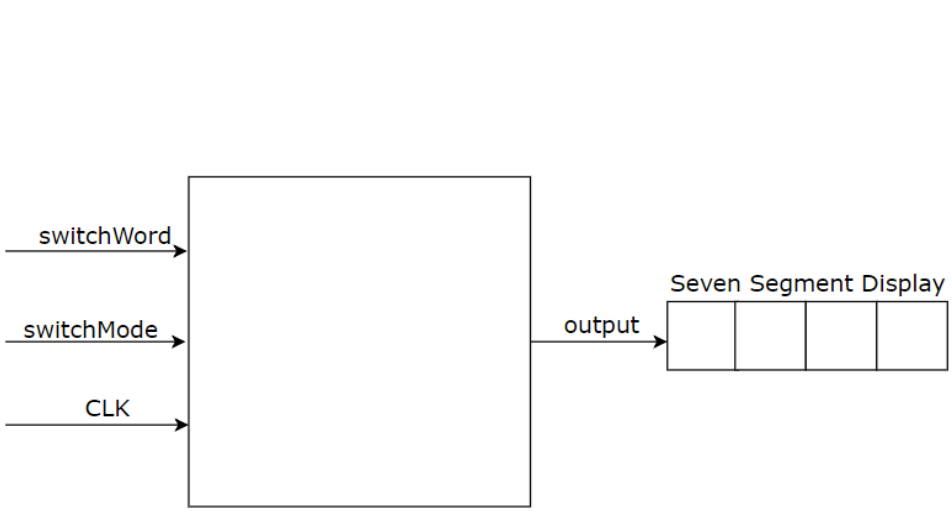
1.2 Enuntul problemei:

A9) Să se proiecteze o reclamă publicitară cu animaţii multiple. Se vor folosi afişajele cu 7 segmente. Textul de afişat va fi format din simboluri ale unui alfabet disponibil. Reclama va avea mai multe regimuri de funcţionare (minimum 4) ce vor putea fi selectate de către utilizator, de la comutatoarele plăcuţei cu FPGA. Se va folosi oscilatorul de cuarţ încorporat în plăcuţa cu FPGA (semnalul de clock respectiv va trebui desigur să fie divizat). Exemple de regimuri de funcţionare: „curgerea" scrisului de la dreapta spre stânga, pâlpâire, afişaj literă cu literă etc.

Deoarece pe un afişaj cu 7 segmente nu se pot reprezenta toate literele, se va crea un alfabet maximal şi mesajele vor fi compuse din simbolurile acelui alfabet. Mesajul va fi conţinut într-o memorie ROM pentru a putea fi uşor schimbat. Proiectul va fi realizat de 1 student.

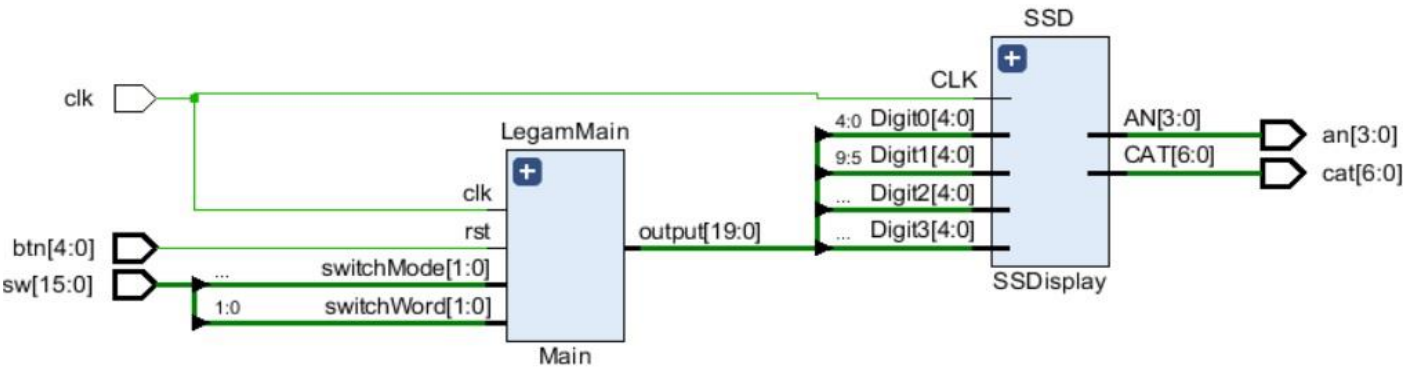
2. Schema bloc a sistemului

a. Black Box

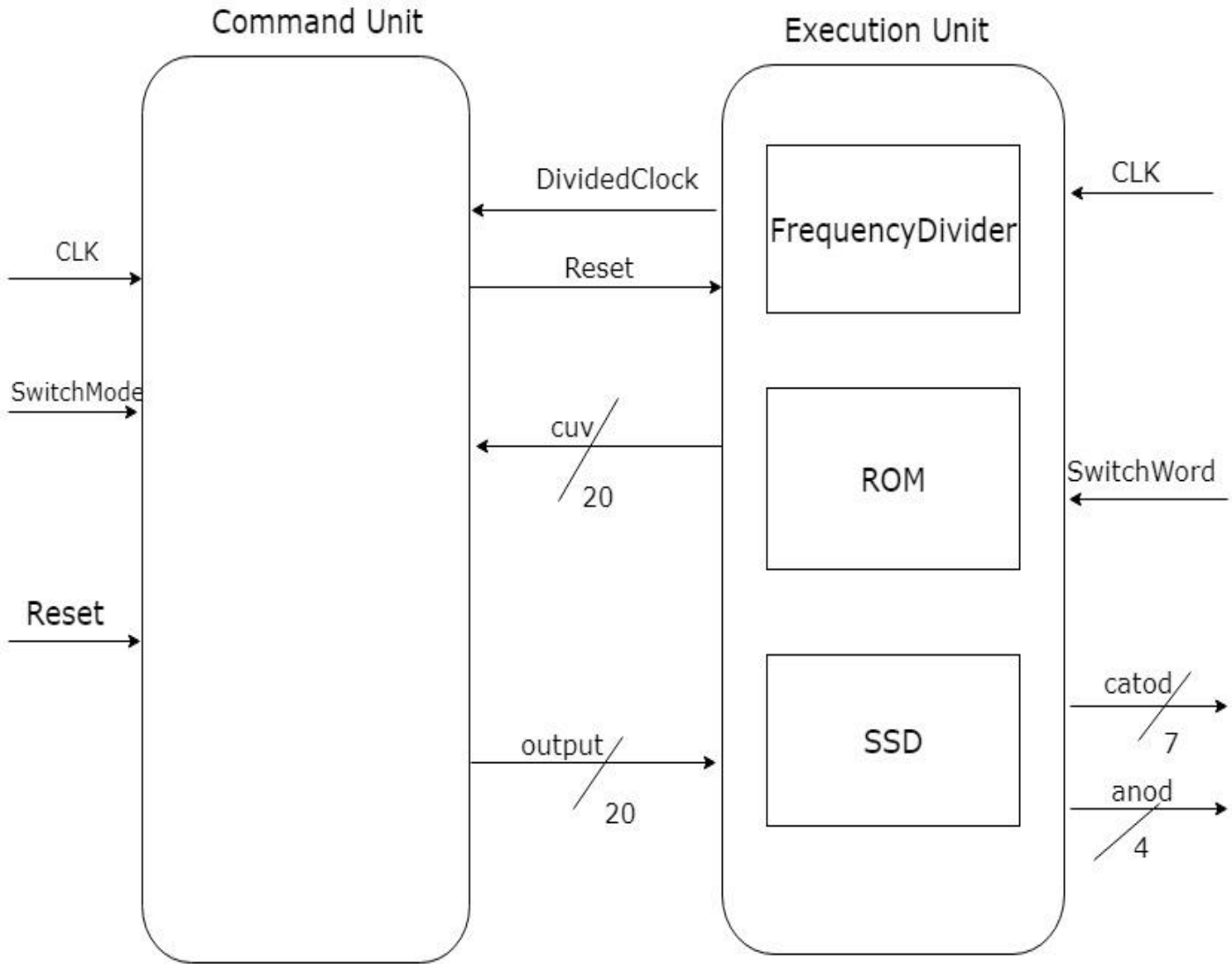


Pentru acest automat avem 3 intrari switchWord, switchMode si CLK. Afișajul se face pe 4 cifre BCD 7 segment.

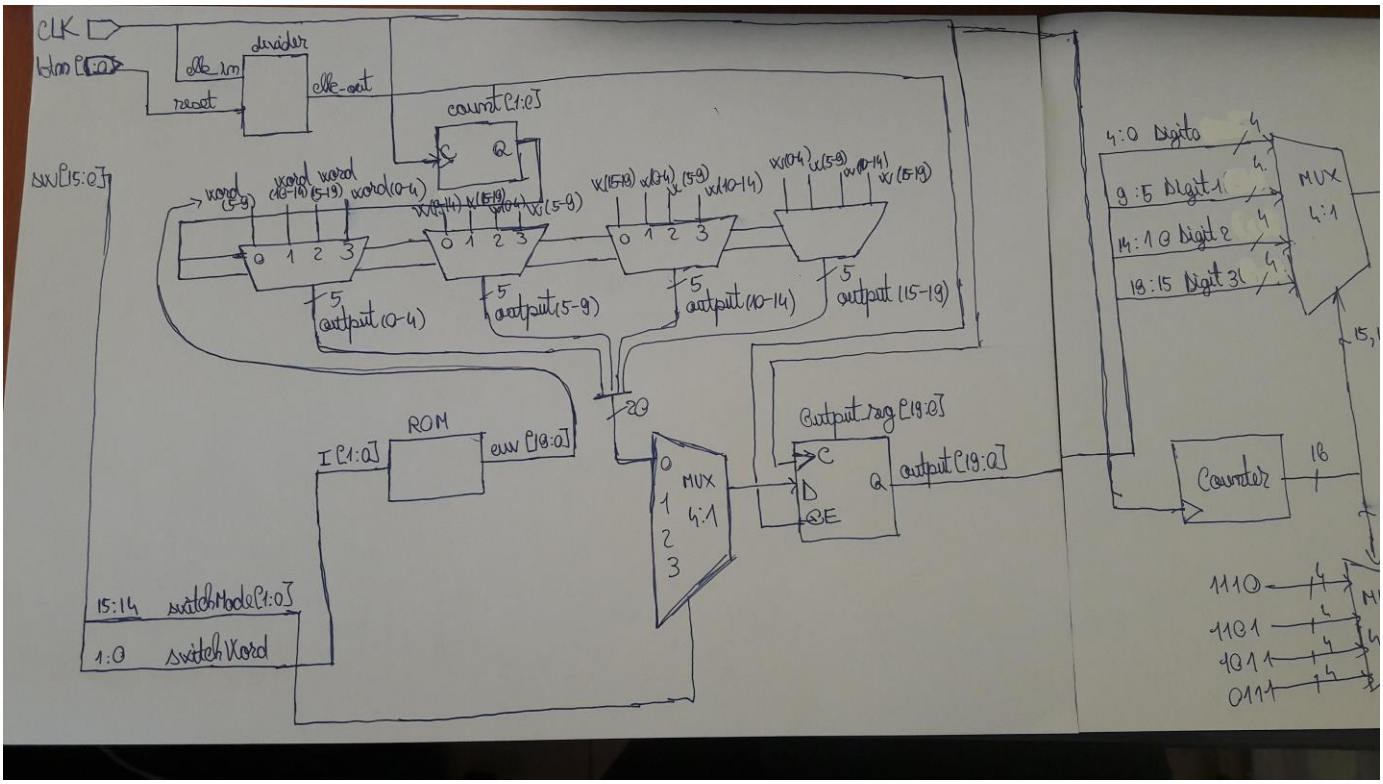
b. Schema bloc

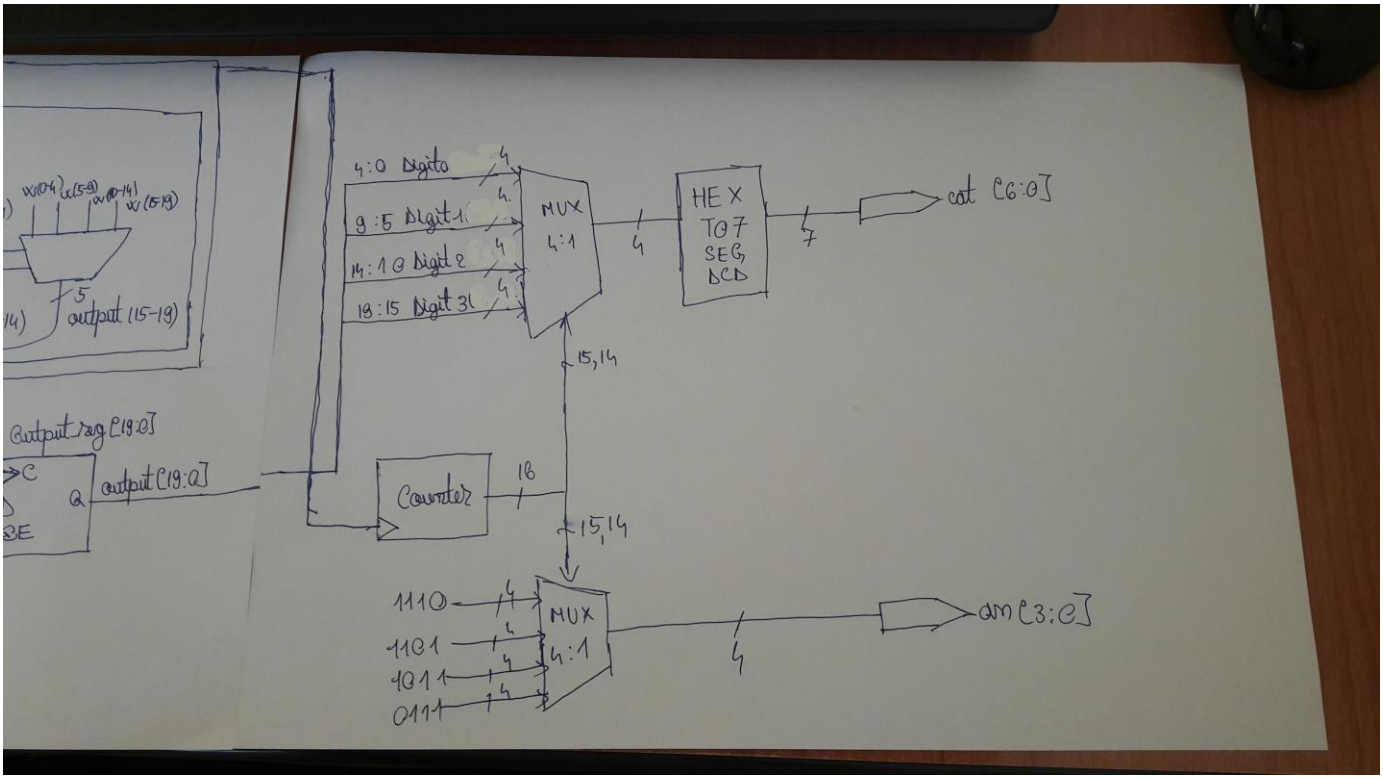


c. Unitatea de comanda si de executie



2.1 Schema bloc detaliata





3.Lista componentelor folosite

1)Cuvintele ce vor fi afisate se regasesc in memoria **ROM.**
Un cuvant are 4 litere, acestea din urma fiind codificate pe 5 biti fiecare. Pentru a schimba cuvintele pe care dorim sa le afisam, ne folosim implicit de adresele diferite la care acestea se regasesc si de 2 comutatoare de pe placuta-SW1 si SW0.
Utilizatorul va putea alege orice text pe care doreste sa-l afiseze, făcând modificări directe in codul VHDL.


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

entity ROM_memory is
port(
  I: in std_logic_vector(1 downto 0); --selectia folosita la switchuri pt a selecta cuvantul
  cuv:buffer std_logic_vector(19 downto 0) --4 litere codificate pe 5 biti
  );
end ROM_memory;

architecture R of ROM_memory is

type mem is array(0 to 3) of std_logic_vector(19 downto 0); --4 cuvinte in memorie a cate 4 litere fiecare, codific

signal my_rom : mem := (
B"00011_01111_01100_00001", --COLA
B"01000_01111_10000_00101", --HOPE
B"00110_10000_00111_00001", --CEAI
B"10100_10011_00011_01110", --UTCN
others => B"00011_00111_01000_00001");

begin

cuv<= my_rom(conv_integer(I)); --efectiv converteste numarul cuvantului aflat in memorie din binar in intreg


end R;
```

2) Afisorul pe 7 segmente cu 4 cifre(SSD-Seven Segment Display)

Placa Basys 3 vine echipata cu un afisor de 7 segmente cu 4 cifre fiecare(Seven Segment Display-SSD). Pe scurt, aceasta interfata foloseste sapte leduri pentru fiecare cifra; fiecare cifra este activata de un semnal de anod. Toate semnalele interfetei SSD(7 semnale comune de catod si 4 semnale distincte de anod) sunt active pe 0 logic. Semnalele de catod controleaza ledurile care se aprind pe acele cifre care au semnalul de anod activ(de exemplu daca se activeaza toate 4 anodurile, atunci se va afisa aceeasi cifra pe cele 4 pozitii).

Pentru a afisa 4 cifre diferite pe SSD, este necesara implementarea unui circuit care trimite cifrele pe semnalele de catod ale SSD in corcondanta cu diagrama de timp de mai jos. Perioada maxima de reimprospatare(refresh) este astfel calculata incat ochiul uman sa nu perceapa aprinderea si stingerea succesiva a fiecarei cifre de pe SSD(16 ms ⇔ 60 Hz). Se realizeaza astfel o afisare ciclica a cifrelor(la un moment dat doar o cifra este afisata, dar ochiul uman nu percepe asta).

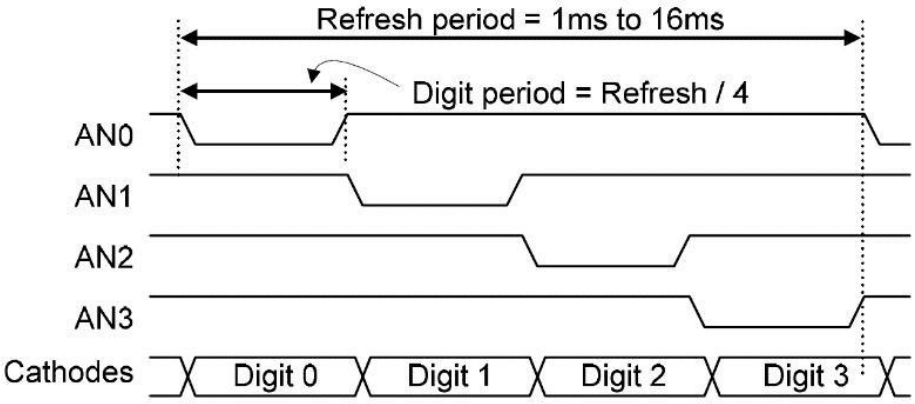
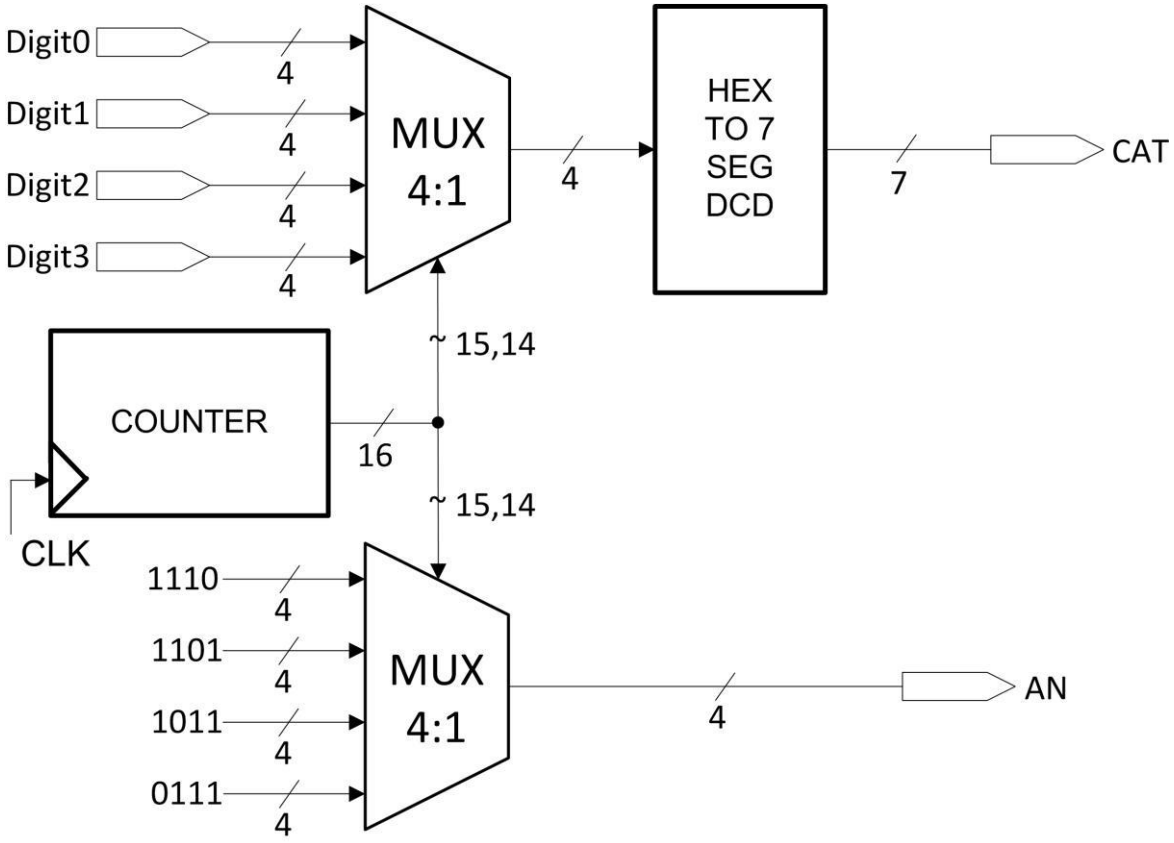


Diagrama de timp pentru SSD

Figura ce urmeaza reprezinta implementarea circuitului de afișare pe SSD. Intrările sunt 4 semnale de 4 biți (cifrele de afișat) și semnalul de ceas al plăcii; ieșirile sunt reprezentate de semnalele de anod (an) și semnalele de catod (cat), toate active pe zero.



Semnalul de ceas are 100 MHz. Folosind biții 15…14 din numărător se va afișa o nouă cifră la fiecare modificare a bitului 14. Bitul 14 se modifică după fiecare 2^{14} cicluri de ceas. Pentru întregul afișor de 4 cifre se obține o frecvență de afișare 100.000.000/(4 * 2^{14})= aproximativ 1500 Hz (mult mai mult decât e necesar).

Tot in SSD codificam si alfabetul utilizat la scrierea cuvintelor:

```
with HEX SELECT

CAT<= "1111111" when "00000",   --SPACE
      "0001000" when "00001",   --A
      "0000011" when "00010",   --b
      "1000110" when "00011",   --C
      "0100001" when "00100",   --d
      "0000110" when "00101",   --E
      "0001110" when "00110",   --F
      "0000010" when "00111",   --G
      "0001001" when "01000",   --H
      "1111001" when "01001",   --I
      "1100001" when "01010",   --J
      "0001111" when "01011",   --k
      "1000111" when "01100",   --L
```

```
"0101010" when "01101",   --m
"0101011" when "01110",   --n
"1000000" when "01111",   --O
"0001100" when "10000",   --P
"0101111" when "10001",   --r
"0010010" when "10010",   --S
"1001110" when "10011",   --T
"1000001" when "10100",   --U
"1100011" when "10101",   --v
"0110110" when "10110",   --x
"0011001" when "10111",   --Y
"0100100" when "11000",   --Z
"1111111" when "11111", -- NONE
"1111111" when others;
```

3)Divizorul de frecventa

Intrucat clock-ul placutei este extrem de rapid, este imposibil ca ochiul uman sa poata percepe afisarea cuvintelor. Astfel, folosim un divizor de frecventa - FrequencyDivider-facut cu cu un numarator, care numara de 100_000_000 de ori semnalul de clock intern al placutei si apoi il schimba. Noul timp dintre doua afisari succesive este acum de aproximativ 2 secunde, facand posibila distingerea clara a continutului.

4)Componenta Main

Un fapt important de retinut este ca in entitatea componentei **Main** vom avea ca porturi declarate switchurile pentru cuvintele de afisat, modurile de afisare ale acestora si un „ouput”, prin intermediul caruia vom afisa cuvantul pe masura ce se modifica.

Memoria ROM si divizorul de frecventa se regasesc ca si componente in Main, ulterior fiind portmapate.

Prelucrarea modurilor de afisare ale cuvintelor este realizata intr-un proces.

```
entity Main is
port(
clk : in std_logic;
switchWord : in std_logic_vector(1 downto 0);
switchMode : in std_logic_vector(1 downto 0);
output : out std_logic_vector(19 downto 0);
rst : in std_logic
);
end Main;

architecture Behavioral of Main is
component ROM_memory is
port(
     I: in std_logic_vector (1 downto 0);
     cuv: buffer std_logic_vector(19 downto 0)
     );
end component;

component clock_divider is
Port (
      clk_in: in STD_LOGIC;
      reset: in STD_LOGIC;
      clk_out: out STD_LOGIC
      );
end component clock_divider;

signal DividedClock : std_logic;
signal word : std_logic_vector(19 downto 0);
signal count : std_logic_vector(1 downto 0) := "00";
signal appear : std_logic := '0';
```

4.1 Multiplexorul 4:1

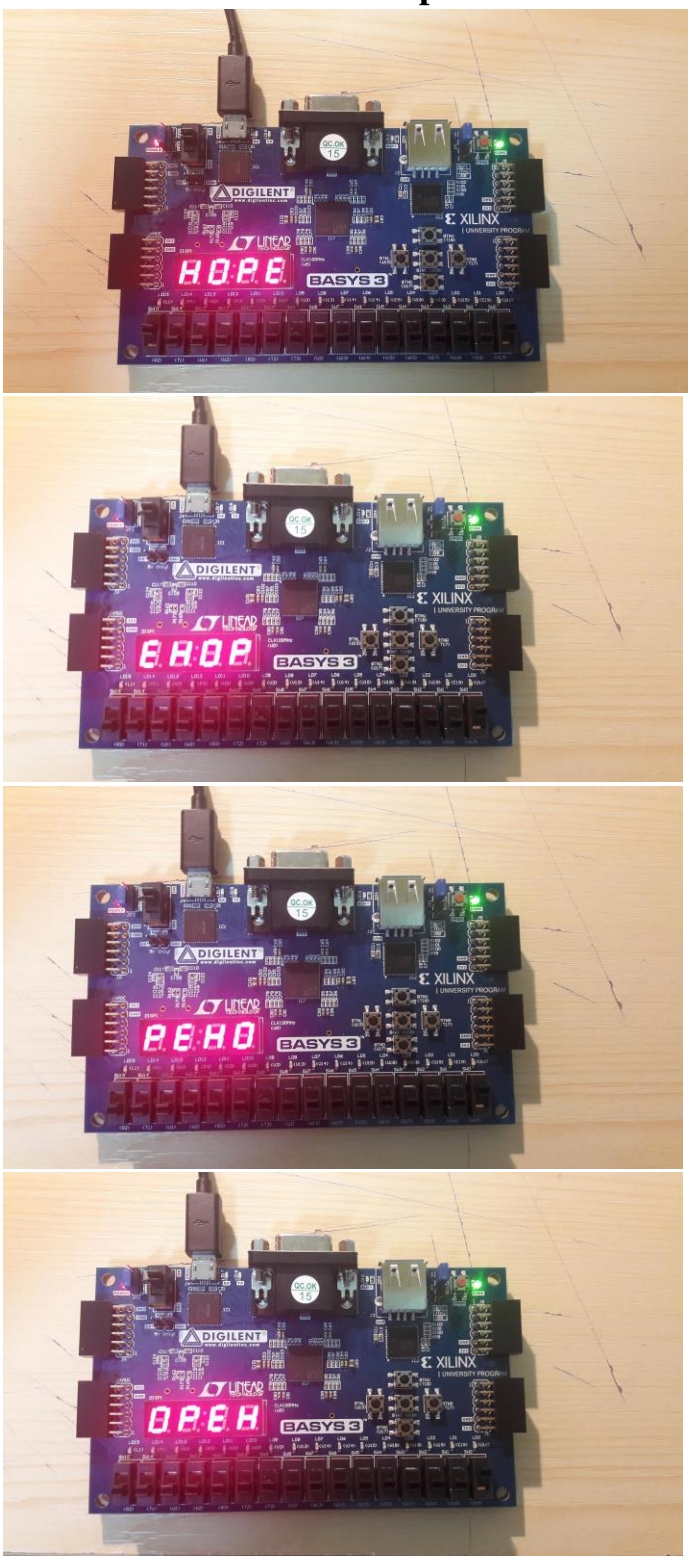
Cu acest multiplexor vom putea alege animația dorită din cele patru.Utilizatorul va putea alege acesta practic prin folosirea a două comutatoare de două poziții din cele 16 disponibile ale placii FPGA (in cazul nostru SW15 si SW14).

4.2 Registrul de deplasare

Registrul de deplasare este folosit pentru a realiza deplasarea literelor textului de afișat intr-o direcție aleasa de către utilizator.

4.3 Animatiile

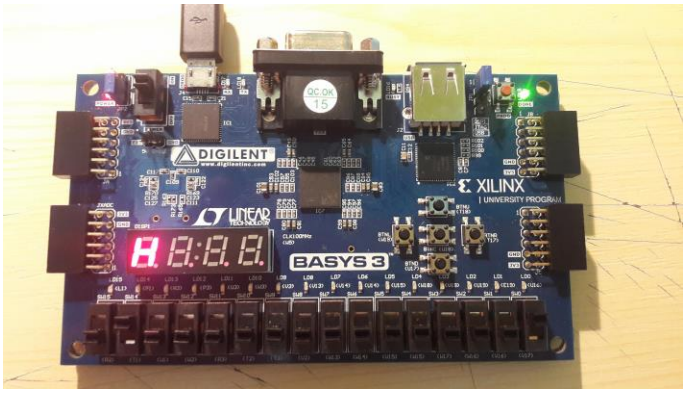
Animatia 1 –Shiftare dreapta

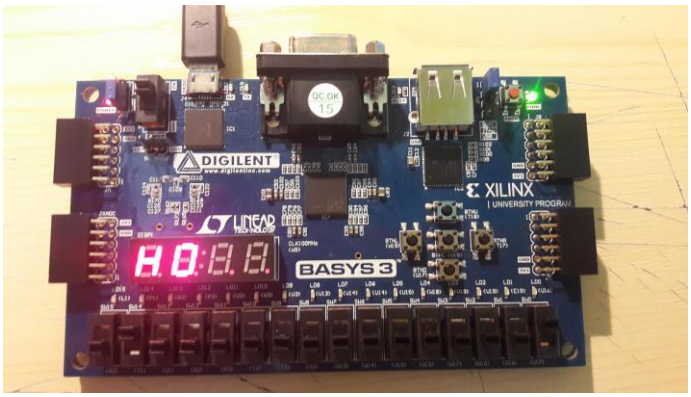


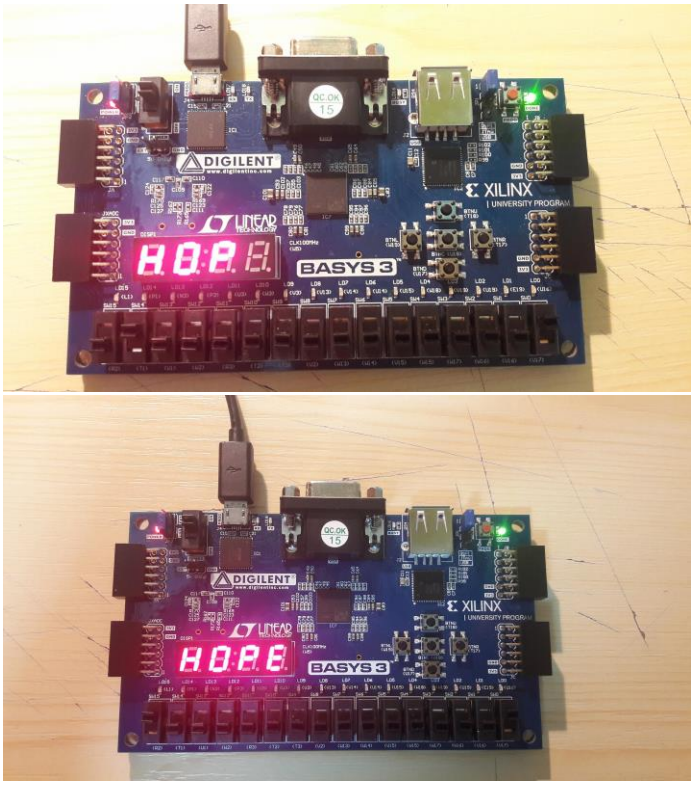
Prima animatie se obine cu ajutorul unui registru de deplasare catre dreapta. O posibila afisare ar fi si afisarea cuvantului afisat catre stanga, ce se poate obtine cu mici modificari in codul primei afisari. Pentru a obține această animație setam valorile pe switch-uri pe 00 logic.

```
43  process(clk, switchMode, DividedClock)
44  begin
45     if ( clk = '1' and clk'EVENT) then
46        if(DividedClock = '1') then
47          case switchMode is
48          -- FIRST APPEARANCE: shift to the right
49              when "00" =>
50                  if(count = "00") then
51                      output(14 downto 10) <= word(19 downto 15);
52                      output(9 downto 5) <= word(14 downto 10);
53                      output(4 downto 0) <= word(9 downto 5);
54                      output(19 downto 15) <= word(4 downto 0);
55                  elsif(count = "01") then
56                      output(14 downto 10) <= word(4 downto 0);
57                      output(9 downto 5) <= word(19 downto 15);
58                      output(4 downto 0) <= word(14 downto 10);
59                      output(19 downto 15) <= word(9 downto 5);
60                  elsif(count = "10") then
61                      output(14 downto 10) <= word(9 downto 5);
62                      output(9 downto 5) <= word(4 downto 0);
63                      output(4 downto 0) <= word(19 downto 15);
64                      output(19 downto 15) <= word(14 downto 10);
65                  else
66                      output(14 downto 10) <= word(14 downto 10);
67                      output(9 downto 5) <= word(9 downto 5);
68                      output(4 downto 0) <= word(4 downto 0);
69                      output(19 downto 15) <= word(19 downto 15);
70                  end if;
```

Animatia 2- Afisare secventiala



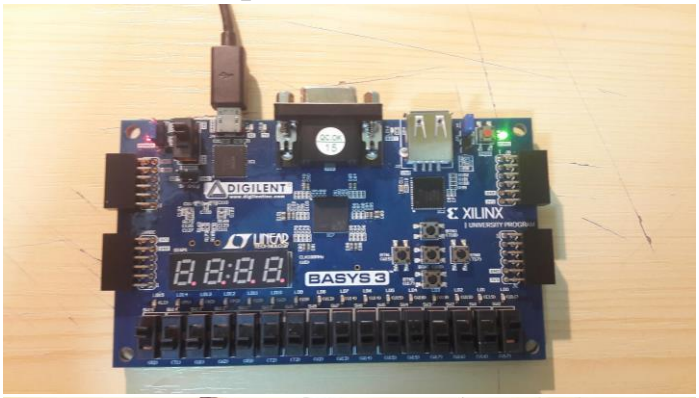




In acest mod de afisare, se incarca secvential in registru cate o litera, fara ca cele precedente ei sa se stinga. Literele care raman stinse, primesc valorea 1 logic pe toti catozii. Pentru a obține această animație setam valorile pe switch-uri pe 01 logic.

```
        --SECOND APPEARANCE
            when "01" =>
                if(count = "00")  then
                    output(19 downto 15) <= word(19 downto 15);
                    output(14 downto 10) <= "11111";
                    output(9 downto 5) <= "11111";
                    output(4 downto 0) <= "11111";
                elsif(count = "01") then
                    output(19 downto 15) <= word(19 downto 15);
                    output(14 downto 10) <= word(14 downto 10);
                    output(9 downto 5) <= "11111";
                    output(4 downto 0) <= "11111";
                elsif(count = "10") then
                    output(19 downto 15) <= word(19 downto 15);
                    output(14 downto 10) <= word(14 downto 10);
                    output(9 downto 5) <= word(9 downto 5);
                    output(4 downto 0) <= "11111";
                else
                    output(19 downto 15) <= word(19 downto 15);
                    output(14 downto 10) <= word(14 downto 10);
                    output(9 downto 5) <= word(9 downto 5);
                    output(4 downto 0) <= word(4 downto 0);
                end if;
```

Animatia 3-Clipire totala

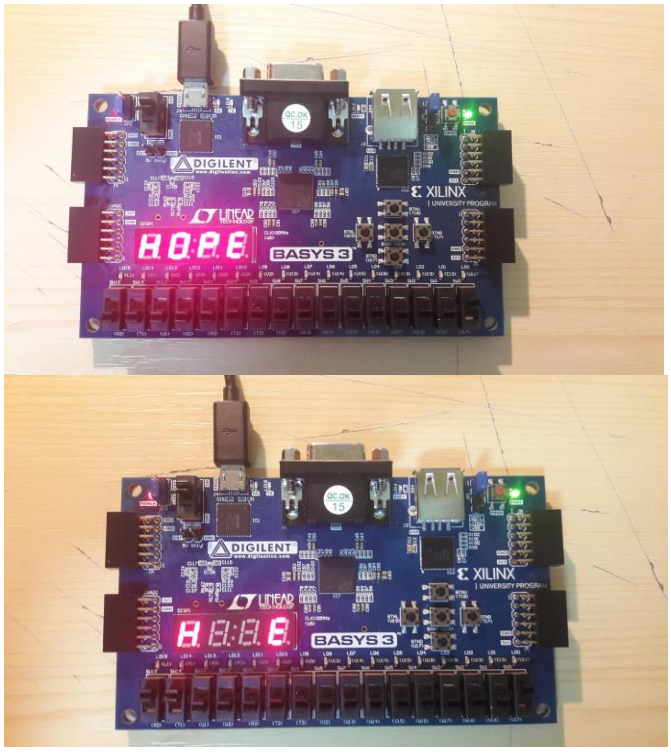


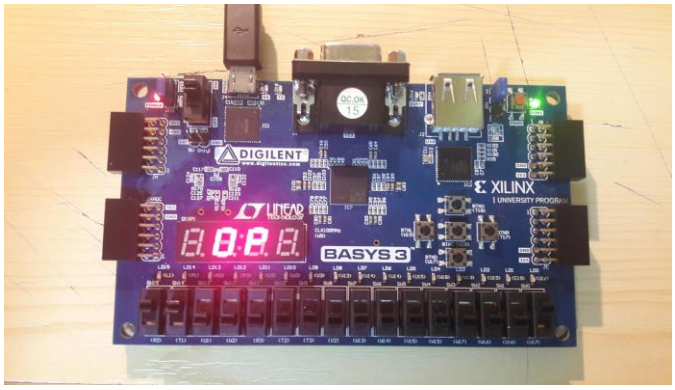
A treia animație poartă numele de -clipire totala-. Textul dorit este

încărcat pe afișoare prin intermediul semnalului „appear” . Cand acesta are valoarea 0, pe placuta nu este afisat nimic, iar cand acesta ia valoarea 1 logic, textul este incarcat complet pe placuta. Pentru a obține această animație schimbăm valorile pe switch-uri pe 10 logic.

```
-- THIRD APPEARANCE: appear/disappear
    when "10" =>
        if appear = '0' then
            output <= X"00000";
            appear <= '1';
        else
            output <= word;
            appear <= '0';
        end if;
```

Animatia 4-afisare cate 2 litere





Cea de-a patra animatie se obtine similar cu cea de-a doua. Incarcam simultan cate 2 litere in registru, celelalte doua nefiind vizibile, intrucat toti catozii acestora au valorea logica 1. Efectul poate fi asemanat cu o imbratisare. Pentru a obține această animație schimbăm valorile pe switch-uri pe 11 logic.

```
104         --FOURTH APPEARANCE: 2 and 2 letters
105           when "11" =>
106               if count = "00" then
107                    output(14 downto 10) <= "00000";
108                    output(9 downto 5) <= "00000";
109                    output(4 downto 0) <= word(4 downto 0);
110                    output(19 downto 15) <= word(19 downto 15);
111               elsif count = "01" then
112                    output(14 downto 10) <= word(14 downto 10);
113                    output(9 downto 5) <= word(9 downto 5);
114                    output(4 downto 0) <= "00000";
115                    output(19 downto 15) <= "00000";
116               else
117                    output(14 downto 10) <= word(14 downto 10);
118                    output(9 downto 5) <= word(9 downto 5);
119                    output(4 downto 0) <= word(4 downto 0);
120                    output(19 downto 15) <= word(19 downto 15);
121               end if;
122           when others =>
123               null;
124       end case;
125       count <= count + 1;
126   end if;
```

5) Componenta „Placuta”

In componenta „placuta” se realizeaza in sine tot proiectul. Componentele „main” si „SSD” apar drept componente la randul lor, fiind ulterior port-mapate. Aici are loc afisarea efectiva si finala a tuturor animatiilor codificate si lucrate pana acum si deci urcarea intregului proiect pe placuta FPGA.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity placuta is
7  port(
8  clk : in STD_LOGIC;
9  btn : in STD_LOGIC_VECTOR(4 downto 0);
10 sw : in STD_LOGIC_VECTOR(15 downto 0);
11 an : out STD_LOGIC_VECTOR(3 downto 0);
12 cat : out STD_LOGIC_VECTOR(6 downto 0)
13 );
14 end placuta;
15
16 architecture Behavioral of placuta is
17
18 component SSDisplay is
19 port(
20 CLK : in std_logic;
21 Digit0 : in std_logic_vector(4 downto 0);
22 Digit1 : in std_logic_vector(4 downto 0);
23 Digit2 : in std_logic_vector(4 downto 0);
24 Digit3 : in std_logic_vector(4 downto 0);
25 AN : out std_logic_vector(3 downto 0);
26 CAT : out std_logic_vector(6 downto 0)
27 );
28
29 end component SSDisplay;
30
31 component Main is
32 port(
33 clk : in std_logic;
34 switchWord : in std_logic_vector(1 downto 0);
35 switchMode : in std_logic_vector(1 downto 0);
36 output : out std_logic_vector(19 downto 0);
37 rst : in std_logic
38 );
39 end component Main;
40
41 signal word : std_logic_vector(19 downto 0);
42
43 begin
44
45
46 LegamMain : Main port map(clk, SW(1 downto 0), SW(15 downto 14), word, BTN(0));
47 SSD : SSDisplay port map(clk, word(4 downto 0), word(9 downto 5), word(14 downto 10), word(19 downto 15), an , CAT);
48
49 end Behavioral;
50
```

4.Constrangeri

clk-> W5 *-clock-ulplacutei*

a-to-g->*-catod*
a_to_g(6)= U7
a_to_g(5)= V5
a_to_g(4)= U5
a_to_g(3)= V8
a_to_g(2)= U8
a_to_g(1)= W6
a_to_g(0)= W7

an->*-anod*
an(3)= W4
an(2)= V4
an(1)= U4
an(0)= U2

SwitchWord->*-alegereacuvantului*
SwitchWord(1)= V16
SwitchWord(0)= V17
SwitchMode->*-alegereamodului de functionare*
SwitchMode(1)= T1
SwitchMode(0)= R2

5. Justificarea solutiei alese

Pentru acest sistem a fost aleasa solutia cea mai simpla si eficienta, intrucat:

- este utilizata o singura memorie ROM
- este utilizat doar clockul de la placuta divizat
- exista un registru in care se retin toate afisarile fiecarei animatii
- afisajul pe 7 segmente este simplu si elegant.

Deoarece a fost un proiect mare, acesta a fost gandit, proiectat si impartit pe componente, fiecare dintre acestea avand o operatie unica.

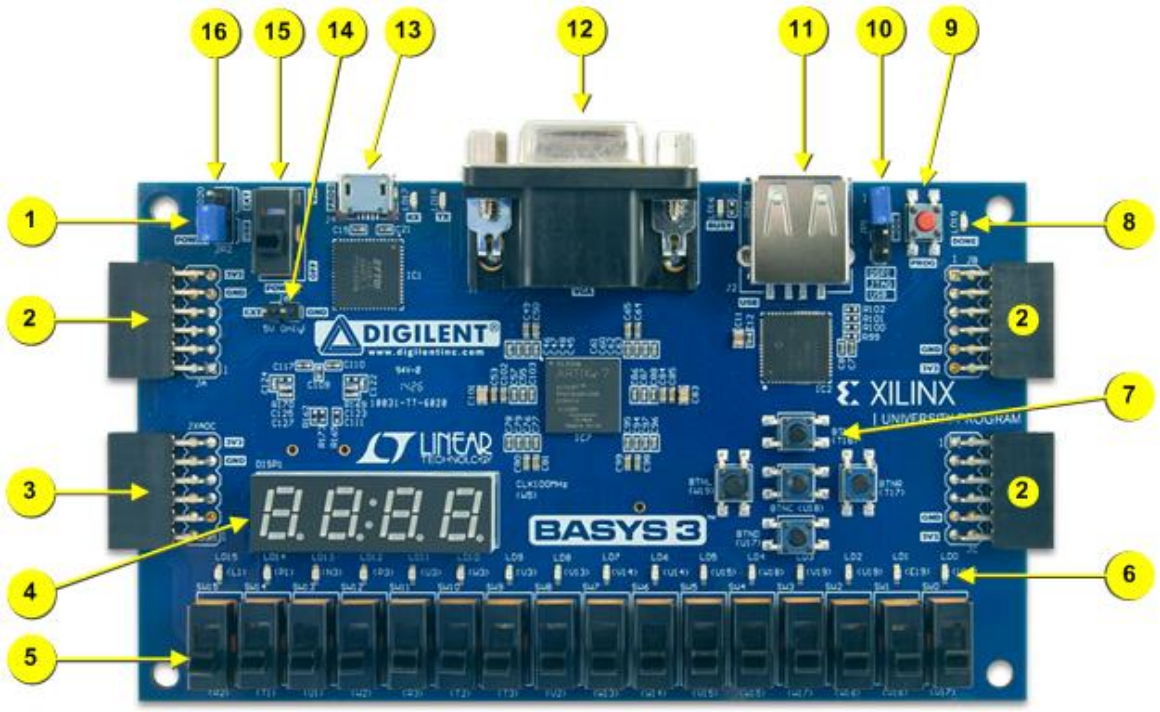
Pentru o urmarire si intelegere eficienta a codului sursa, au fost alese denumiri simple si semnificative, prescurtari sau chiar nume complete, sugestive. De asemenea, au fost adaugate comentarii la proiect.

Toate acestea justifica alegerea acestei solutii teoretic si practic eficiente.

6. Posibilități de dezvoltare ulterioară

- Introducerea de cuvinte de mai multe litere
- Animații mai dinamice care să atragă atenția asupra reclamei

7. Instructiuni de utilizare



La inceputul proiectului, tuturor switchurilor utilizate de pe placuta (SW15, SW14, SW1,SW0) le este atribuita valoarea „00” logic. Primul cuvant din memoria ROM este incarcat si afisat pe SSD, in primul mod de animatie- shiftare dreapta.

De fiecare data cand dorim sa vedem un cu totul alt cuvant din memorie, selectam o valoare din intervalul 0-3, codificata in binar, de pe SW1 SI SW0. Putem alege astfel orice cuvant, indiferent de ordinea adreselor din memoria ROM.

Cuvantului ales ii putem selecta oricand unul din cele 4 moduri de afisare, atribuind SW15 si SW14 o valoare din intervalul 0-3, la randul sau codificata in binar.

In acest fel, pe afisorul cu 7 segmente al placutei va fi mereu un cuvant afisat intr-un anumit mod, nefiind intreruperi sau pauze nejustificate.