Reclamă publicitară cu animații multiple

Boca Ioan-Doru

Grupa 30214

Contents

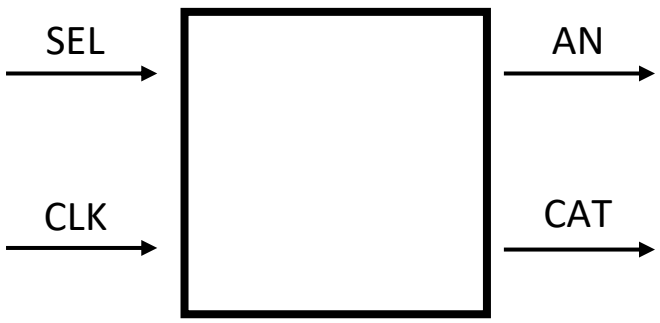
1. Specificații

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 pentru a putea fi uşor de schimbat.

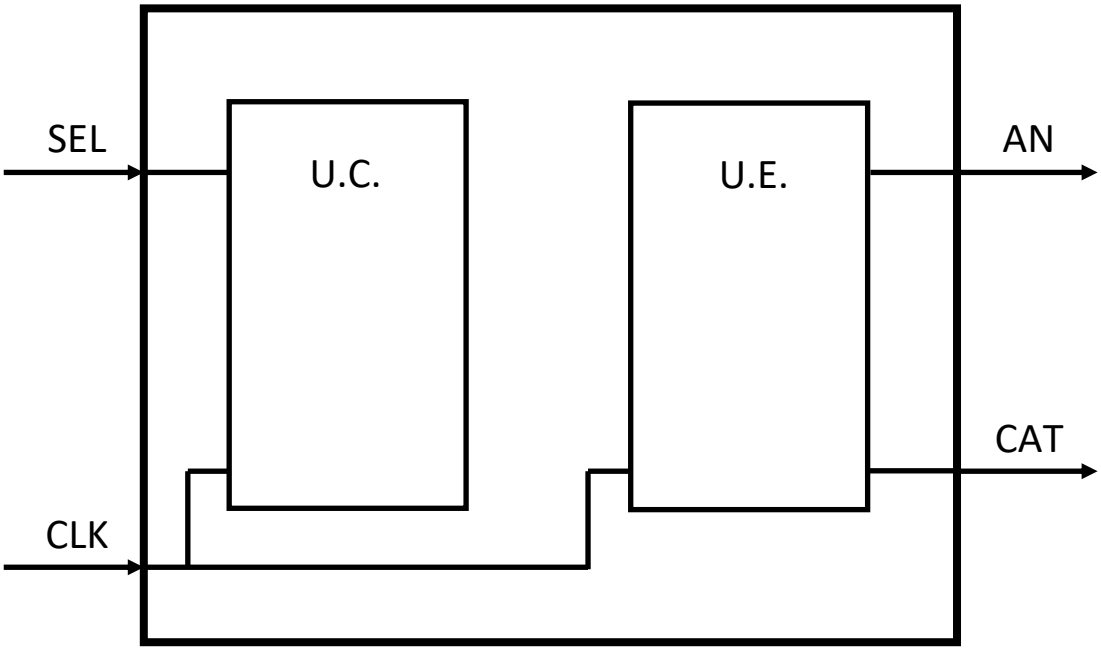
2. Proiectare

2.1 Schema bloc



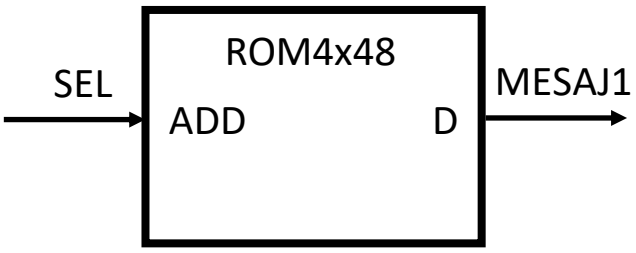
2.2 Unitatea de control și Unitatea de execuție

2.2.1 Maparea intrarilor și a ieșirilor



2.2.2 Resurse U.C.

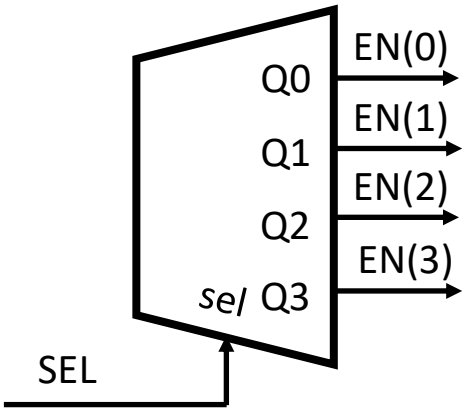
1. ROM4x48



Aceasta este o memorie ROM cu adrese de 2 biți și ieșire pe 48 biți. Este folosit pentru stocarea mesajelor codificate: 8 caractere cu dimensiune de 6 biți. Mesajul este ales în funcție de animația selectată.

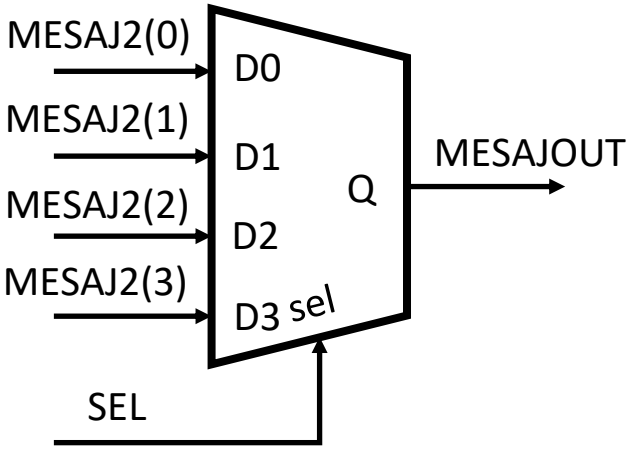
ADD	D	Mesaj decodificat
00	0x31730A31754A	COCACOLA
01	0x31730A31754A	COCACOLA
10	0x8E49669E8A6A	Simboluri speciale
11	0xAEBAEBAEBAEB	--------

2. DMUX1x4



Acesta este un DMUX cu selecția pe 2 biți și ieșiri de 1 bit. Este folosit pentru activarea animației selectate și dezactivarea celorlalte. Face legătura dintre UC și UE prin semnalul de ieșire EN folosit pentru start/reset la animații.

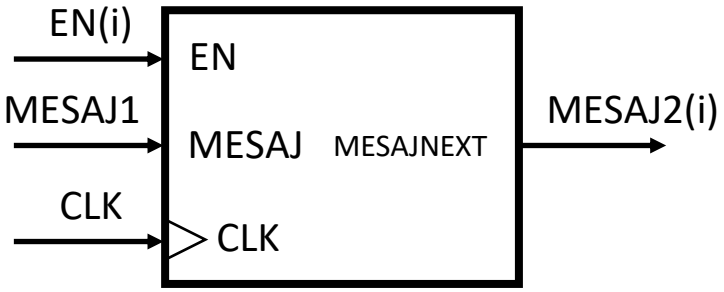
3. MUX4x1



Acesta este un MUX cu selecția pe 2 biți și intrari/ieșiri de 48 biți. Mesajul este generat la fiecare frame al animației în UE, iar mesajul de ieșire este ales în funcție de animația selectată.

2.2.3 Resurse U.E.

1. Animații



Schema bloc a animațiilor este identică, diferențele sunt la nivel intern, fiecare animație funcționând pe principii diferite. Intrarea EN specifica starea activă a animației, astfel dacă EN='1' animația funcționează, iar pentru EN='0' animația este în starea de reset. Resetarea are loc în momentul selecției unei alte animații prin resetarea numărătorului intern și a semnalului de ceas, MESAJNEXT primește valoarea xFFFFFFFFFFFF(8 caractere spațiu). Clock-ul intern al componentei diferă în funcție de fiecare animație. Ieșirea MESAJNEXT specifică mesajul ce va fi afișat în frame-ul următor.

i. ANIMATIE1

Frecvența semnalului de ceas este de 1Hz, se face o shiftare la stânga cu un caracter(6 biți) a mesajului de intrare. În momentul în care tot mesajul a fost afișat pe cele 8 segmente

shiftarea se produce cu 6 biți de ‘1’(codul “111111” reprezintă caracterul spațiu).

```
process(clkD2,EN)
begin
if EN='0' then
    char_counter <= x"0";
    mesaj2<= x"FFFFFFFFFFF";
elsif rising_edge(clkD2) then
    if char_counter(3)='0' then
        mesaj2<=mesaj2(41 downto 0)&MESAJ((7-conv_integer(char_counter))*6+5 downto (7-conv_integer(char_counter))*6);
    else
        mesaj2<=mesaj2(41 downto 0)&"111111";
    end if;
    char_counter <= char_counter + 1;
end if;
MESAJNEXT<= mesaj2;
end process;
end Behavioral;
```

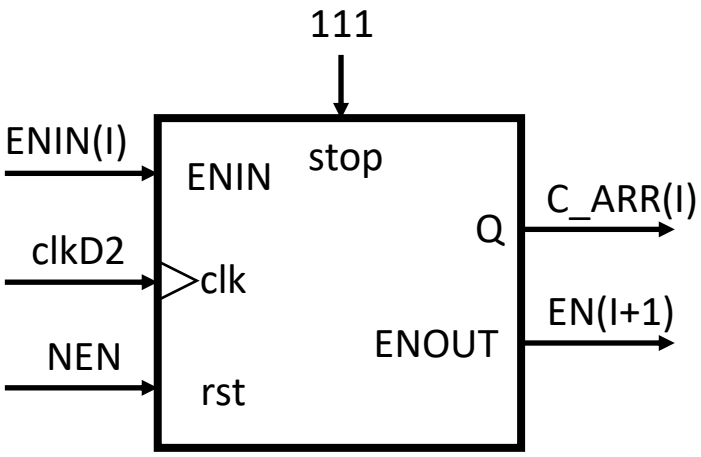
ii. ANIMATIE2

Frecvența semnalului de ceas este de 1Hz, animația constă în a face mesajul să pâlpâie. Astfel la un semnal de ceas MESAJNEXT primește MESAJ, iar la următorul valoarea xFFFFFFFFFFF.

```
process(clkD2,EN)
begin
if EN='0' then
    n<='0';
    MESAJNEXT<= x"FFFFFFFFFFF";
elsif rising_edge(clkD2) then
    if n='1' then
        MESAJNEXT<=MESAJ;
        n<='0';
    else
        MESAJNEXT<=x"FFFFFFFFFFF";
        n<='1';
    end if;
end if;
end process;
end Behavioral;
```

iii. ANIMATIE3

Frecvența semnalului de ceas este de 4Hz, aceasta este o animație de loading. Sunt folosite 8 numărătoare modulo 8 cascadate prin intrarea ENIN și ieșirea ENOUT:



- ENIN -valoarea de enable a numărătorului
- rst - resetează numărătorul la 0 si ENOUT=0
- Q -ieșirea pe 3 biți care reprezintă numărul
- stop -valoare generică
- ENOUT -ieșirea pentru activarea numărătorului următor, se activează când Q=stop-1

Fiecare caracter i din MESAJNEXT primește caracterul de pe poziția C_ARR(i) dacă ENIN(i)=1 sau “111111” dacă ENIN(i)=0.

```
67
68  NEN<=not(EN);
69  CLKDIVC2: clock_div generic map(12_500_000) port map(CLK,NEN,clkD2);--12_500_000
70
71  ANIMLOAD:for I in 0 to 7 generate
72      ALOAD: COUNTER generic map("111") port map(ENIN(I),clkD2,NEN,C_ARR(I),ENIN(I+1));
73  end generate ANIMLOAD;
74
75  process(clkD2,EN)
76  begin
77  if EN='0' then
78      MESAJNEXT<= x"FFFFFFFFFFFF";
79  elsif rising_edge(clkD2) then
80      for I in 0 to 7 loop
81          if ENIN(I)='1' then
82              MESAJNEXT(I*6+5 downto I*6)<=MESAJ(conv_integer(C_ARR(I))*6+5 downto conv_integer(C_ARR(I))*6);
83          else
84              MESAJNEXT(I*6+5 downto I*6)<="111111";
85          end if;
86      end loop;
87  end if;
88  end process;
89  end Behavioral;
90
```

iv. ANIMATIE4

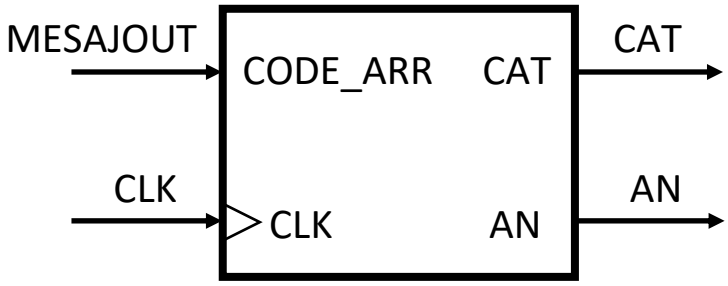
Frecvența semnalului de ceas este variabilă între 0.25Hz și 2Hz, aceasta este o animație de loading. Se încarcă pe rând caracterele pe MESAJNEXT, în momentul în care toate cele 8

caractere au fost încărcate mesajul se resetează și animația reincepe.

```
process(CLK,EN)
begin
if EN='0' then
    count<=1;
    tmp<='0';
elsif(rising_edge(CLK)) then
    count <=count+1;
    if (count = N/(conv_integer(char_counter)+1)) then
        tmp <= NOT tmp;
        count <= 1;
    end if;
end if;
clkD2 <= tmp;
end process;

process(clkD2,EN)
begin
if EN='0' then
    mesaj2<=x"FFFFFFFFFFFF";
    char_counter<="0000";
elsif rising_edge(clkD2) then
    mesaj2<=mesaj2(41 downto 0)&MESAJ((7-conv_integer(char_counter))*6+5 downto (7-conv_integer(char_counter))*6);
    char_counter <= char_counter + 1;
    if char_counter="1000" then
        char_counter<="0000";
        mesaj2<=x"FFFFFFFFFFFF";
    end if;
end if;
MESAJNEXT<= mesaj2;
end process;
end Behavioral;
```

2. SEVENSEGMENT



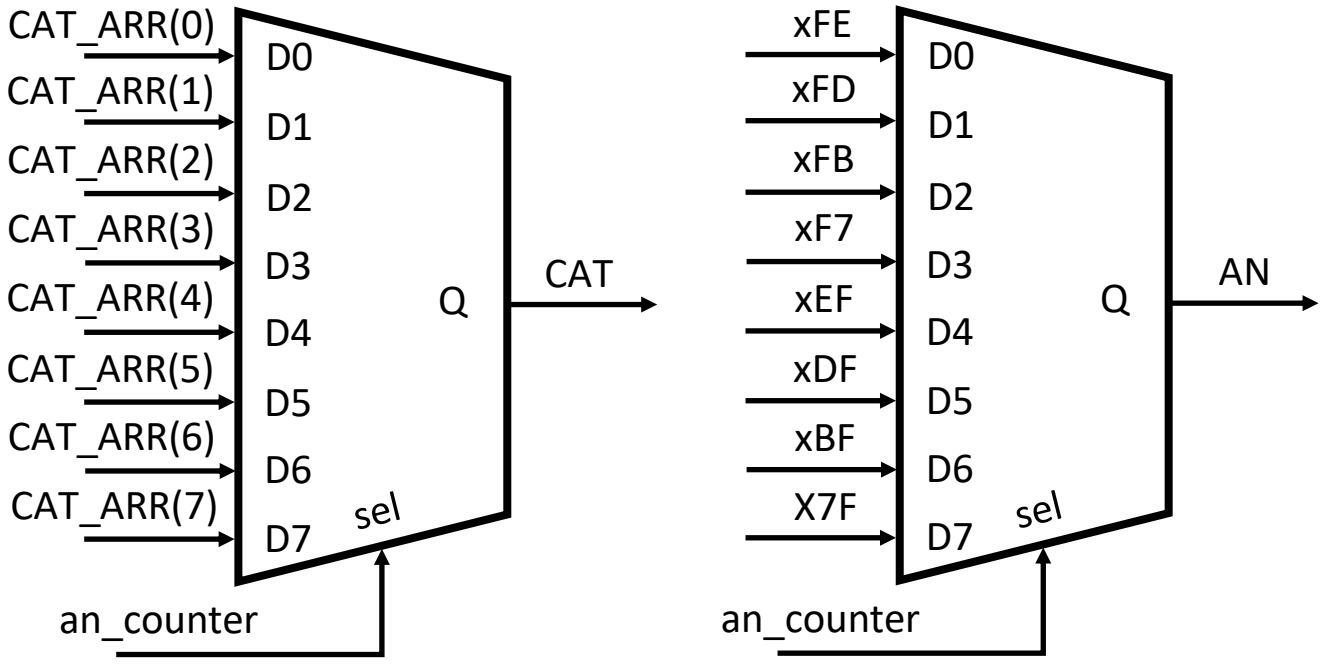
Această componentă efectuează transformarea codurilor mesajului în ieșirile pentru catozi și afișarea acestora. Refresh rate-ul este de 60Hz. Codul fiecărui caracter este pus în semnalul CHAR_ARR de tipul CODE_Array. Sunt folosite mai multe componente:

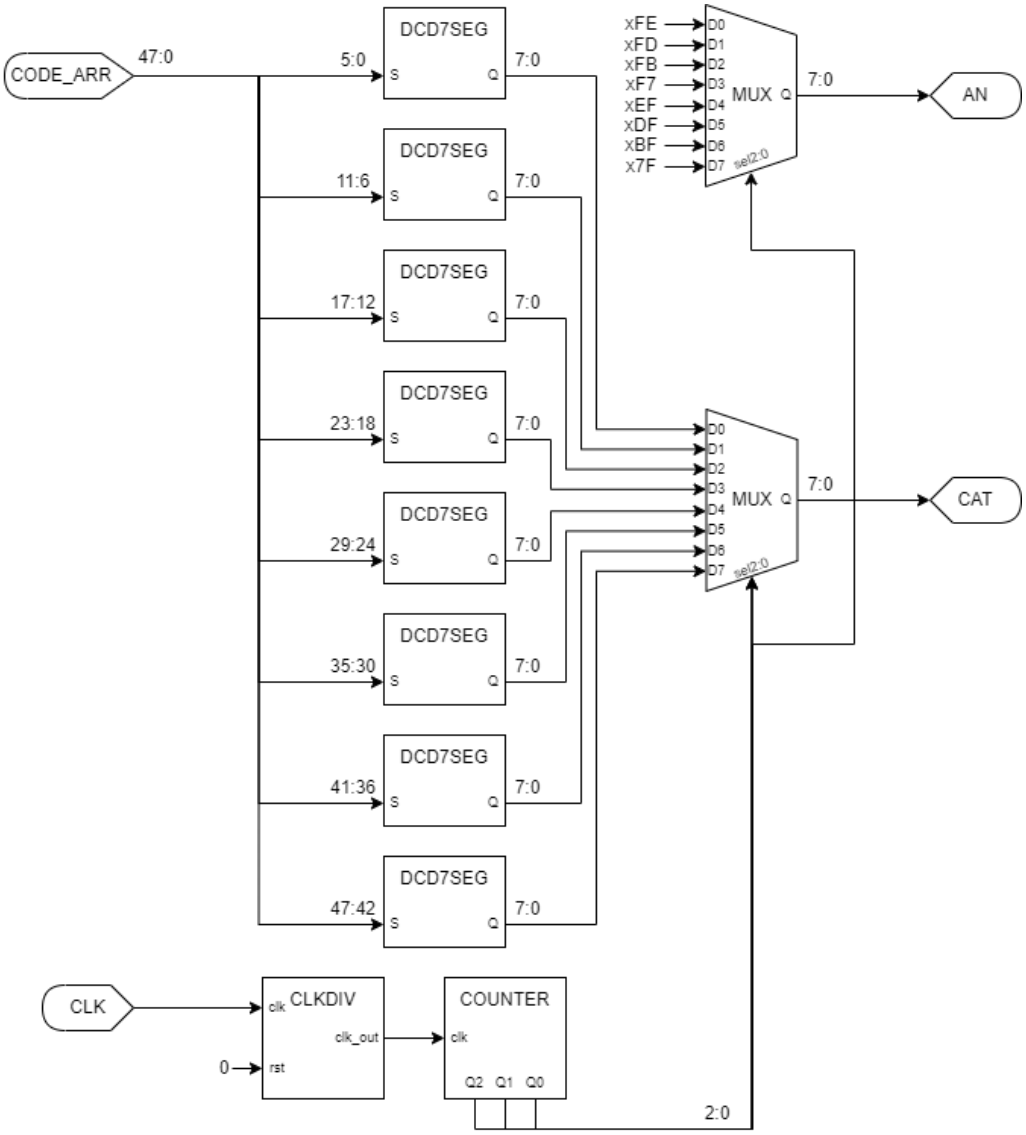
i. DCD7SEG

Pentru fiecare caracter se află valoarea pentru catozi. Este folosit un alfabet maximal care contine cifrele 0-9, literele alfabetului latin în afară de {M, V, W, Z}, caracterele speciale ‘.’, ‘!’, ‘?’ și 8 caractere personalizate folosite pentru a treia animație.

ii. MUX8x1

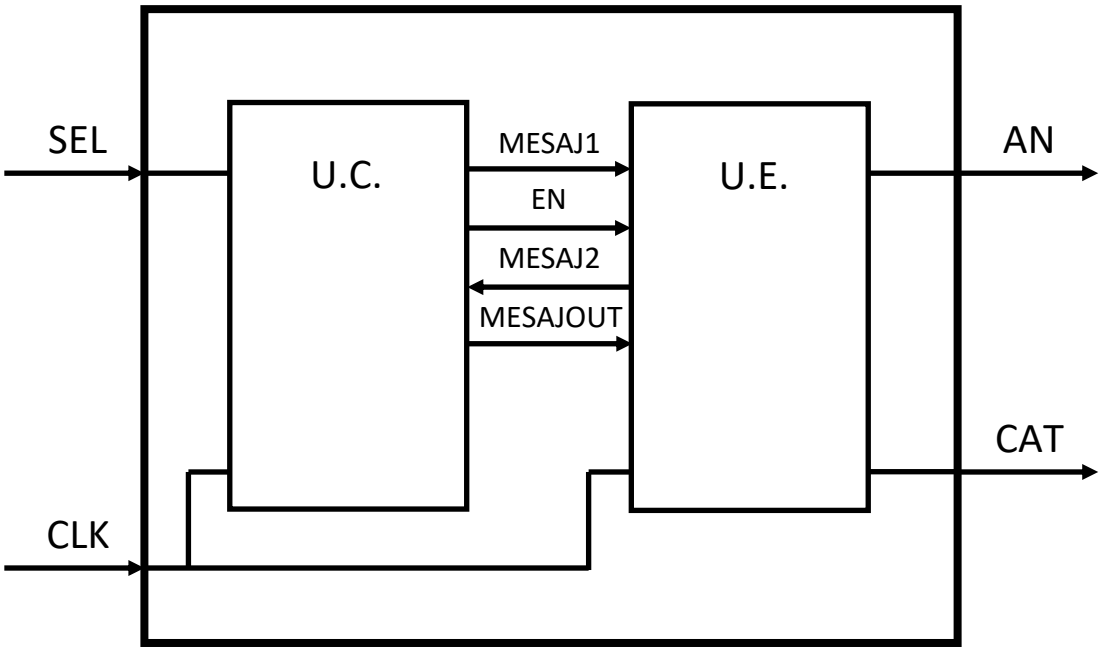
Sunt folosite 2 componente MUX8x1 cu intrari și ieșiri de 8 biți. Semnalul an_counter, folosit pentru selecție, este ieșirea unui numărător intern modulo 8. Primul este folosit pentru a alege valoarea potirvita pentru ieșirea CAT în funcție de selecția an_counter care reprezintă afisorul curent. Al doilea este folosit pentru a activa un singur anod pe rand(acestea fiind active pe 0, valorile hexazecimale folosite au un singur bit de 0).



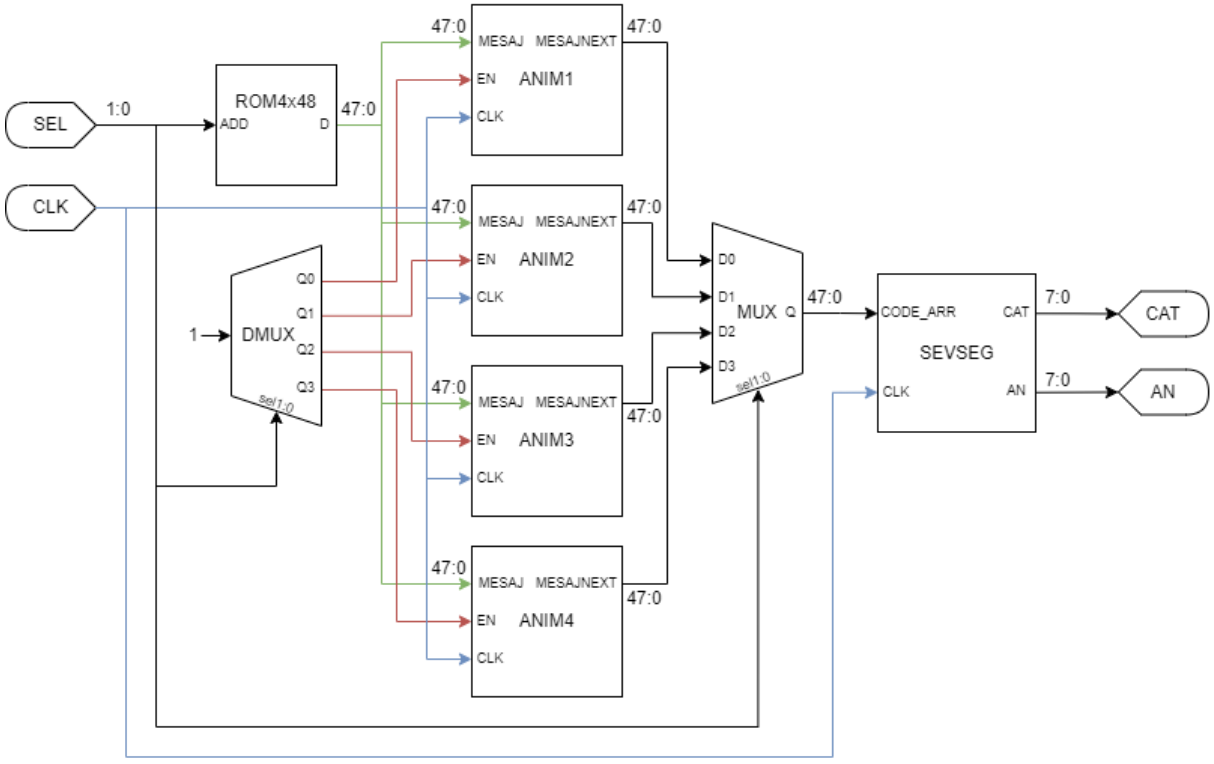


Schema completă a circuitului SEVENSEGMENT

2.2.4 Schema bloc cu legăturile dintre UC si UE



2.2.5 Schema in detaliu a proiectului



3. Manual de utilizare si intreținere

Selecția animațiilor se face de la primele doua switch-uri astfel:

- Pentru “00” se va afișa animația 1 “mers de la dreapta la stânga”
- Pentru “01” se va afișa animația 2 “pâlpâie”
- Pentru “10” se va afișa animația 3 “loading prin cascadare”
- Pentru “11” se va afișa animația 4 “loading cu timp variabil”

4. Justificarea soluției alese

Am ales această metodă pentru eficiența din punct de vedere a memoriei și simplitatea codului, fiind ușor modificabil pentru viitoare versiuni. Mesajul reclamei poate avea doar 8 caractere deoarece animația 2 ar necesita un algoritm de descompunere în subșiruri de maxim 8 caractere a mesajului de afișat. Inca un motiv ar fi necesitatea de a parametriza marea majoritate a componentelor pentru a susține un mesaj cu număr variabil de caractere.

Pentru componenta SEVENSEGMENT am ales folosirea a 8 componente DCD pentru o traducere mai usoară a codurilor în caractere afișabile, o metoda ce ar putea fi implementată pentru a folosi mai puține componente este de a traduce codul caracterului doar în momentul afișării acestuia.

5. Posibilitați de dezvoltari ulterioare

Versiunile următoare pot introduce:

- Mai multe animații folosind metoda prezentata prin componente specifice animației dorite pentru implementare.

- Posibilitatea afișării unui text de lungime variabilă prin mărirea memoriei pentru mesaje, parametrizarea componentelor utilizate și implementarea unui algoritm de descompunere a mesajului în subșiruri.

6. Bibliografie

- Îndrumătorul de laborator
- Suportul de curs
- Manualul de utilizare VHDL
- Manualul de referință Nexys A7
- Stackoverflow