*Reclama publicitara cu animatii multiple*

*Facultatea de Automatica si Calculatoare*

*Sectia Calculatoare si Tehnologia Informatiei*

*Student: Bumbu Iulia-Diana*

*An 1, seria B, grupa 30218*

*Profesor indrumator: Pop Diana*

***Cuprins:***

1. *Specificatia proiectului*
   1. *Studiul domeniului tehnic*
      1. *Enunt*
2. *Proiectarea*

*2.1 Analiza functionala a sistemului*

*2.1.1 Diagrama UML*

*2.2 Schema bloc*

*2.2.1 Schema*

*2.2.2 Intrari/iesiri*

*2.3 Descrierea formala a componentelor*

1. *Sinteza*

*3.1 Codificarea starilor logice*

1. *Justificarea solutiei alese*
2. *Instructiuni de utilizare si intretinere*
3. *Posibilitati de dezvoltare ulterioare*

*1)Specificatia proiectului*

* 1. *Studiul domeniului tehnic*

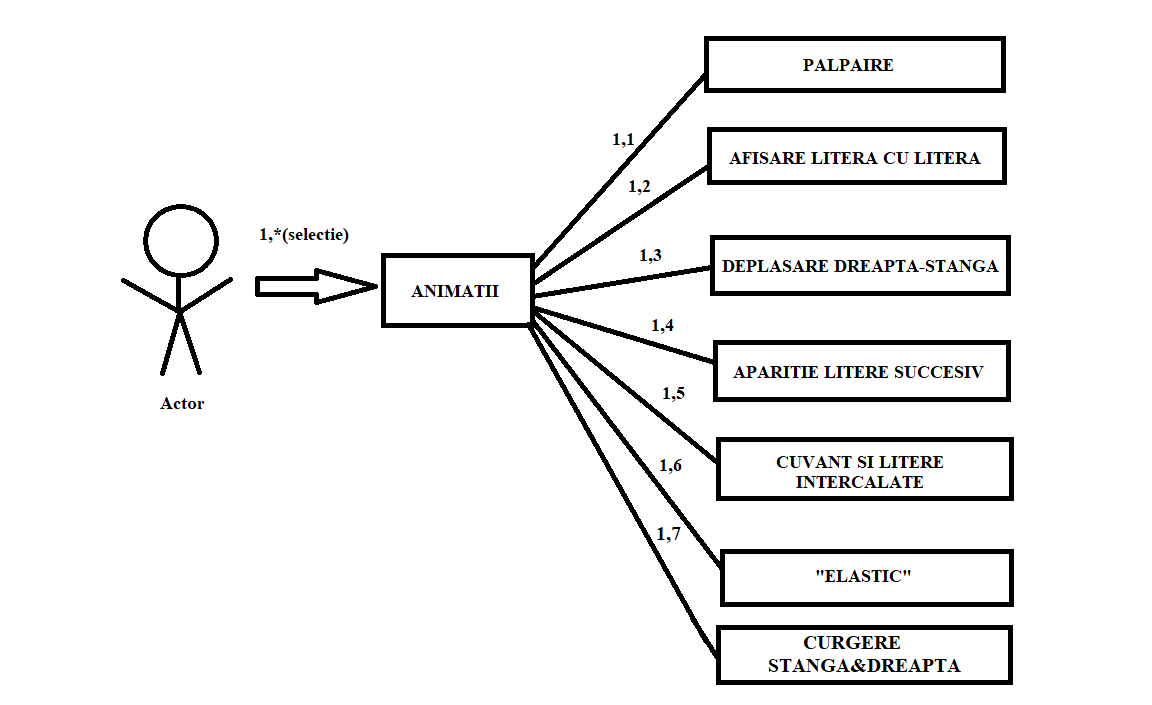
*1.1.1 Enunt*

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 (minimum 4) ce vor putea fi selectate de catre utilizator de la comutatoarele placutei cu FPGA. Se va folosi oscilatorul de cuart incorporat in placuta cu FPGA (semnalul de clock respectiv va trebui desigur sa fie divizat). Exemple de regimuri de functionare: “curgerea” scrisului de la dreapta spre stanga, palpaire, afisaj litera cu litera etc. Deoarece pe un afisaj cu 7 segmente nu se pot reprezenta toate literele se va crea un alfabet maximal si mesajele vor fi compuse din simbolurile acelui alfabet. Mesajul poate fi continut intr-o memorie ROM pentru a putea fi usor schimbat. Proiectul va fi realizat de 1 student.

*2) Proiectarea*

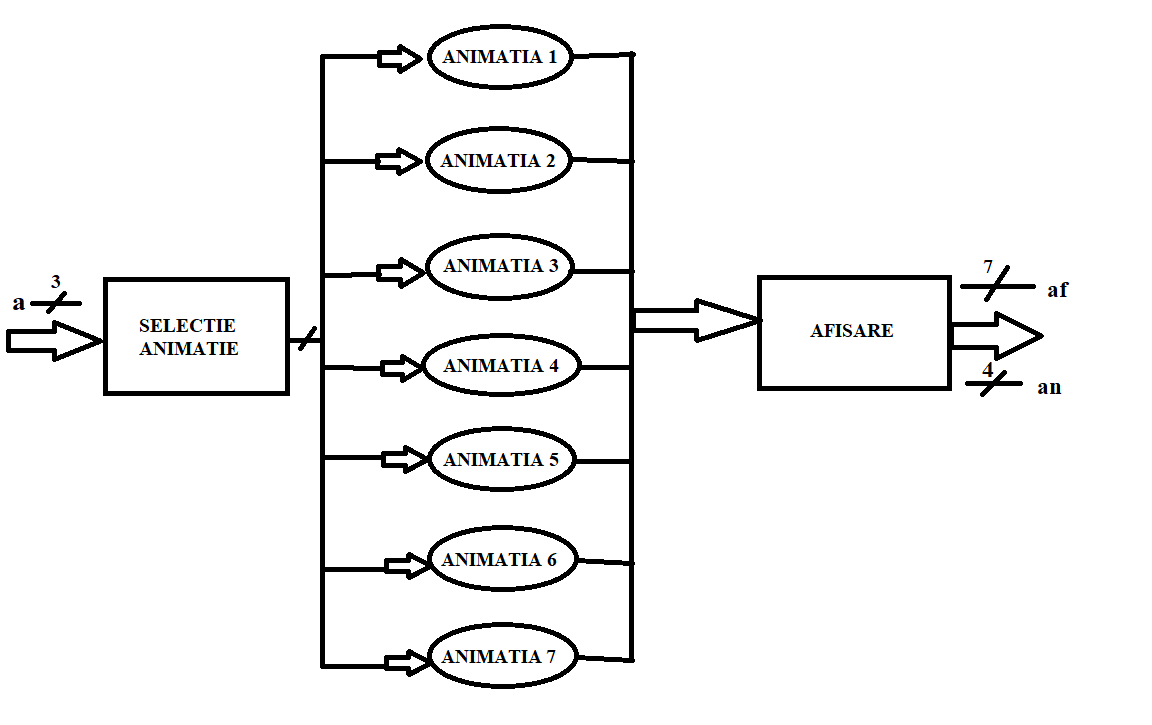
*2.1 Analiza functionala a sistemului*

*2.1.1 Diagrama UML*

****

*2.2 Schema bloc*

*2.2.1 Schema*

****

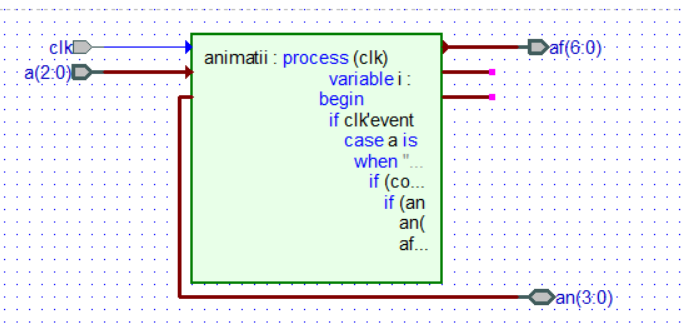
*2.2.2 Intrari/Iesiri*

**a:** tipin, cale de date pe 3 biti, permite utilizatorului sa selecteze animatia dorita:

**an**:tip inout, cale de date pe 4 biti, permite controlarea celor 4 anoduri

**af**:tip out, cale de date pe 7 biti, permite activarea celor 7 segmente ale afisorului

**clk**:tip in, cale de date pe un bit, reprezinta semnalul de clock de la placuta



* 1. *Descrierea formala a componentelor:*

Tabelul de adevar al alfabetului maximal creat:

| ***Litera*** | ***a*** | ***b*** | ***c*** | ***d*** | ***e*** | ***f*** | ***g*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *spatiu* | *1* | *1* | *1* | *1* | *1* | *1* | *1* |
| *A* | *0* | *0* | *0* | *1* | *0* | *0* | *0* |
| *B* | *1* | *1* | *0* | *0* | *0* | *0* | *0* |
| *C* | *0* | *1* | *1* | *0* | *0* | *0* | *1* |
| *D* | *1* | *0* | *0* | *0* | *0* | *1* | *0* |
| *E* | *0* | *1* | *1* | *0* | *0* | *0* | *0* |
| *F* | *0* | *1* | *1* | *1* | *0* | *0* | *0* |
| *G* | *0* | *1* | *0* | *0* | *0* | *0* | *0* |
| *H* | *1* | *1* | *0* | *1* | *0* | *0* | *0* |
| *I* | *1* | *1* | *1* | *1* | *0* | *0* | *1* |
| *J* | *1* | *0* | *0* | *0* | *1* | *1* | *1* |
| *K* | *1* | *1* | *1* | *1* | *0* | *0* | *1* |
| *L* | *1* | *1* | *1* | *0* | *0* | *0* | *1* |
| *M* | *0* | *1* | *0* | *1* | *0* | *1* | *0* |
| *N* | *1* | *1* | *0* | *1* | *0* | *1* | *0* |
| *O* | *0* | *0* | *0* | *0* | *0* | *0* | *1* |
| *P* | *0* | *0* | *1* | *1* | *0* | *0* | *0* |
| *R* | *1* | *1* | *1* | *1* | *0* | *1* | *0* |
| *S* | *0* | *1* | *0* | *0* | *1* | *0* | *0* |
| *T* | *0* | *1* | *1* | *1* | *0* | *0* | *1* |
| *U* | *1* | *0* | *0* | *0* | *0* | *0* | *1* |
| *V* | *1* | *1* | *0* | *0* | *0* | *1* | *1* |
| *W* | *1* | *0* | *0* | *0* | *0* | *0* | *0* |
| *X* | *1* | *0* | *0* | *1* | *0* | *0* | *0* |
| *Y* | *1* | *0* | *0* | *1* | *1* | *0* | *0* |
| *Z* | *0* | *0* | *1* | *0* | *0* | *1* | *0* |

**

Alte notatii existente in cod:

* count: (signal count: std\_logic\_vector(26 downto 0):=(others=>'0');)

-placuta utilizata are un clock care functioneaza la 100 MHz, deci avem nevoie de o modalitate de a incetini executia pentru a putea fi vizibila animatia de ochiul uman

- in cod acest semnal are rolul de a intarzia modificarea variabilei “i” cu aproximativ o secunda: 10^8~2^27, astfel fiecare faza fiind observabila pe afisor in acest interval de timp

* count1:( signal count1: std\_logic\_vector(25 downto 0):=(others=>'0');)

-este folosit in cadrul animatiei 6 (“elastic”) cu rol de a intarzia modificarea variabilei “i” cu mai putin de o secunda pentru a asigura rapiditatea executiei

* count\_a: (signal count\_a: std\_logic\_vector(16 downto 0):=(others=>'0');)

-asigura periodizarea si secventierea corecta a comenzilor pentru anozi si segmente, deoarece frecventa oscilatorului placii este mult prea mare pentru a genera periodicizarea corespunzatoare

-frecventa de improspatare a informatiei pentru fiecare anod trebuie sa fie intre 1KHz si 60Hz pentru ca ochiul uman sa nu perceapa inchiderea si deschiderea anozilor si sa aiba „iluzia” ca pe fiecare anod apare o litera diferita concomitent

- de fiecare data cand semnalul ajunge la valoarea 2^17 se va schimba litera care trebuie afisata precum si anodul pe care trebuie sa apara

-valoarea ‘17’ a fost aleasa conform calculelor:

100Mhz/2^17~763 Hz

1/763~0.0013=1,3 msec => fiecare litera va fi aprinsa pt cca 13 msec, perioada de improspatare devenind 4\*1,3=5,2 msec care apartine (1,16)msec

* i: (variable i: natural:=0;)

- va creste cu 4 unitati aprox din secunda in secunda asigurand incarcarea urmatoarelor 4 constante

-in cazul in care ajunge la valoarea maxim posibila corespunzatoare animatiei respective variabila va lua valoarea 0, reincepand bucla

* alfabet: (type alfabet is array (25 downto 0) of std\_logic\_vector (6 downto 0);)

-contine cele 26 de constante corespunzatoare alfabetului maximal din tabelul anterior

* l: (constant l: alfabet:=(....)’ )

-vectorul de litere asociat alfabetului

* anim i: (type anim i is array (x downto 0) of std\_logic\_vector (6 downto 0);)

-contine “fazele” animatiei i, i=1-7 adica constantele corespunzatoare codificarii literelor/spatiilor care constituie respectivul mod de functionare

-x variaza in functie de cate faze contine animatia respective

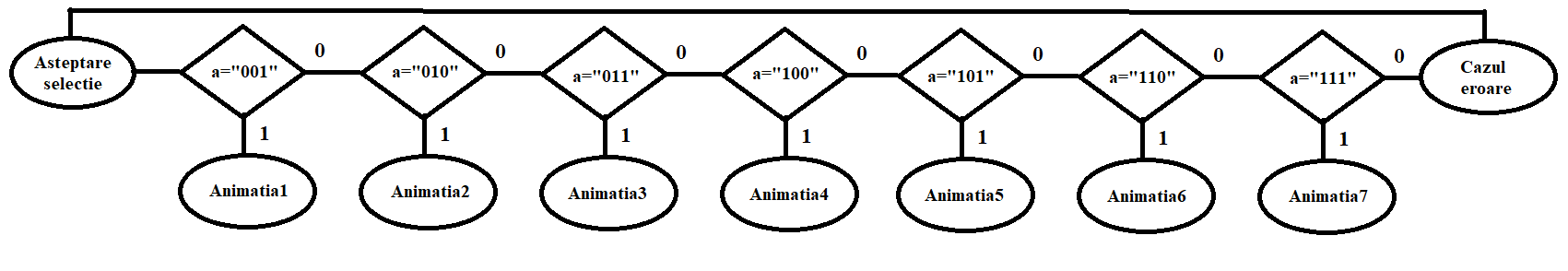
* ci: (constant ci: anim i:=(…..); )

-contine constantele asociate fiecarei animatii, i=1-7

*3) Sinteza*

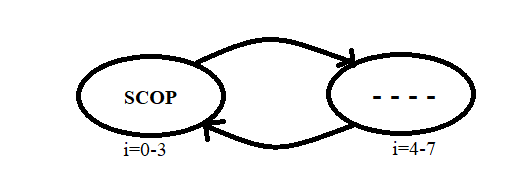
*3.1 Codificarea starilor logice*

Reclama contine 7 stari (cele 6 animatii+mesajul de eroare) . Cuvantul ales este „SCOP”.

**

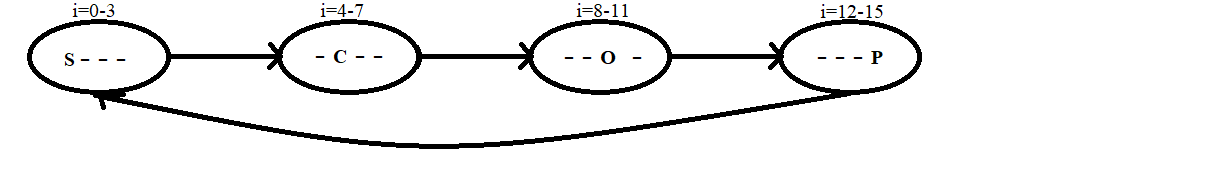
**Animatia1**(palpaire)

Contine cele 4 litere si 4 spatii:



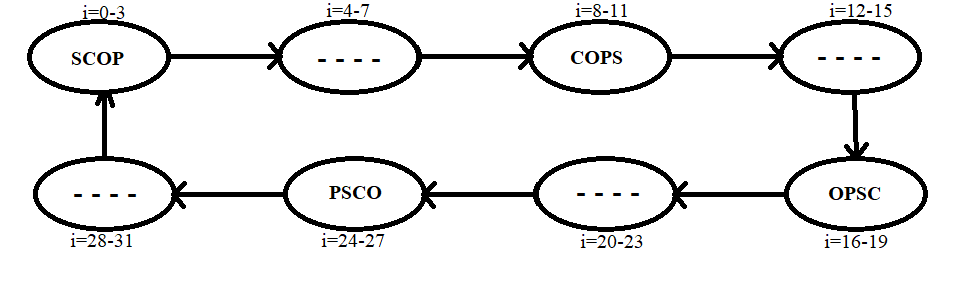
**Animatia2**(afisare litera cu litera)

Contine cele fiecare dintre cele 4 litere si cate 3 spatii:



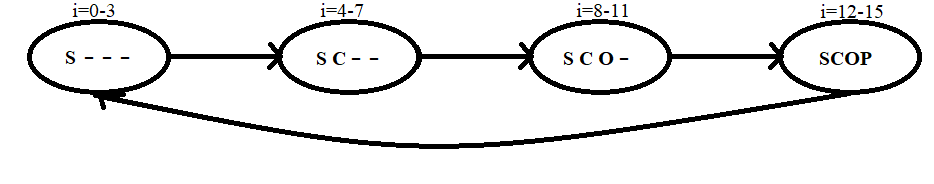
**Animatia3**(deplasare dreapta-stanga)

Contine permutarile corespondente celor 4 litere precum si cate 4 spatii intre ele:



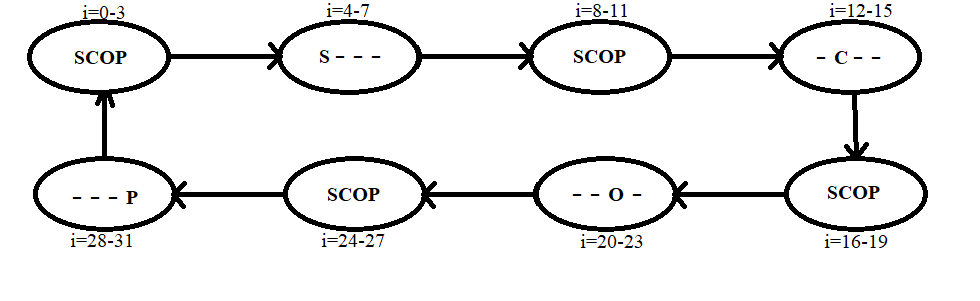
**Animatia4**(aparitie litere succesiv)

Contine cele i litere si 4-i spatii, i=1-4:



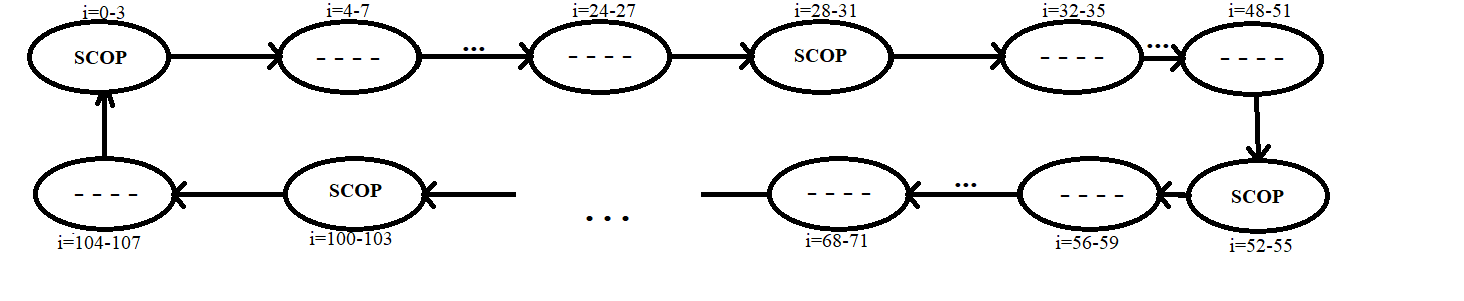
**Animatia5**(afisare cuvant si literele constituente intercalat)

Contine litere si spatii conform codificarii:



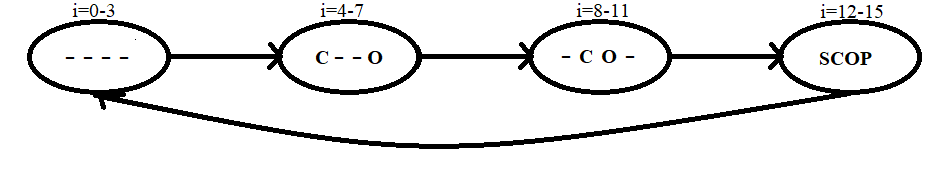
**Animatia6**(„elastic”)

Contine cele 4 litere si un nr variabil de spatii:



**Animatia7**(curgere din stanga si din dreapta)

Are urmatoarea configuratie:



**Cazul de eroare**

Pe afisor va fi vazut mesajul **EEEE**

*4) Justificarea solutiei alese*

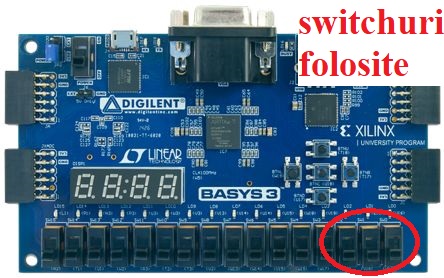
Am decis sa abordez acest proiect din punct de vedere comportamental deoarece acest tip de descriere hardware este asemanator limbajelor de programare clasice, programul avand astfel structura unui „algoritm”. Sistemul este modelat la un nivel inalt de abstractizare, principala preocupare fiind functionalitatea modelului descris. Lizibilitatea codului este asigurata si de manipularea animatiilor in cadrul unui proces deoarece gandirea secventiala este mult mai usor de inteles decat cea concurenta, inclusiv pentru un utilizator nefamiliarizat cu acest domeniu.

*5)Instructiuni de utilizare si intretinere*

Reclama are disponibile 7 animatii codificate conform tabelului de mai jos. In cazul in care se alege o codificare pentru care nu exista nicio animatie ( 000 ) pe afisor va aparea mesajul de eroare “EEEE”. 3 comutatoare de 2 pozitii (intrerupatoare) sunt accesibile utilizatorului pentru a introduce informatie pentru a genera semnale de intrare. Cand un comutator este pozitionat pe UP sau ON, este transmisa starea High(deci 1 logic) respectiv starea Low(deci 0 logic) cand comutatorul este pozitionat pe DOWN sau OFF.

Utilizatorul trebuie sa controleze cele 3 comutatoare conform codificarii date pentru a observa pe afisorul placutei animatia dorita.

|  |  |
| --- | --- |
| **a[2],a[1],a[0]** | **REGIMURI** |
| 001 | Palpaire |
| 010 | Afisare litera cu litera |
| 011 | Deplasare dreapta stanga |
| 100 | Aparitie litere succesiv |
| 101 | Cuvant si litere intercalate |
| 110 | „Elastic” |
| 111 | Curgere din stanga&dreapta |
| 000 | Mesaj de eroare |



*6)Posibilitati de dezvoltare ulterioare*

* Utilizatorul poate alege sa afiseze alt cuvant facand modificari in codul **vhdl** prin schimbarea valorii vectorului

**ci(), i=1-7**. Litera corespunzatoare mesajului dorit se alege din cadrul familiei de vectori de litere **l()**.

* Este posibila completarea alfabetului maximal cu alte notatii precum si cu numere sau diversi operatori, adaugandu-se valori in vectorul de constante l().
* O alta imbunatatire adusa proiectului poate fi stocarea unui text mai lung decat cel deja existent ( 4 litere). In acest caz afisarea acestuia se va face gradat, cate 4 litere odata, din cauza faptului ca sunt disponibile doar 4 afisoare BCD 7 segmente.
* Se pot implementa alte animatii mai dinamice. Daca avem mai mult de 8 moduri de functionare intrarea **a** va avea calea de date pe minim 4 biti. Pentru fiecare animatie nou implementata se va crea un nou tip:

type anim\_i is array (x downto 0) of std\_logic\_vector (6 downto 0);

precum si un vecor ci() de constante: constant ci: anim\_i:=( …..));