



Structura sistemelor de calcul: folosirea VGA pentru a afisa o imagine pe monitor si procesare de imagini

Proiect de laborator

Nume:Aghenitei Bianca si Serbu Cristian

Grupa:30235

Teaching Assistant: Alexandru Dan Butiri



Contents

1 Rezumat	3
2 Introducere	4
3 Fundamentare teoretica	5
3.1 Reprezentarea imaginilor	5
3.2 Interfata VGA	5
3.3 Procesarea imaginilor	7
3.4 Filtrul Greyscale	7
3.5 Filtrul Binary	7
3.6 Filtrul Edge Detection	8
4 Proiectare si implementare	10
4.1 Driver VGA	10
4.2 Stocarea imaginii	14
4.3 Filtrul greyscale si binary	16
4.4 Filtrul edge detection	17
4.5 Diagrama RTL	21
4.6 Manual de utilizare	21
5 Rezultate experimentale	22
5.1 Testare	22
5.2 Dificultati intalnite	24
6 Concluzii	26

Chapter 1

Rezumat

Problema abordata consta in afisarea pe un ecran a unei imagini prin portul VGA si prelucrarea acesteia folosind filtre de image processing. Obiectivele principale sunt proiectarea unui driver pentru afisarea unei imagini folosind interfata VGA si definirea filtrelor pentru procesarea imaginilor.

Implementarea va fi facuta pe placuta Basys3, ce va fi programata folosind limbajul de programare hardware VHDL. Imaginile vor fi stocate in format .coe pe blockul de memorie RAM integrat al placutei.

Implementarea a fost dusa cu succes pana la capat, iar rezultatele sunt cele asteptate: imaginea este afisata corect pe ecran, iarfiltrele sunt aplicate corespunzator, fara a distorsiona imaginea.

Chapter 2

Introducere

In cadrul acestui proiect vom incerca vom afisa imagini pe ecran folosind portul VGA al placii Basys3. Un conector Video Graphics Array (VGA) este un conector standard utilizat pentru ieșirea video a computerului. Originar din 1984, conectorul cu 15 pini a devenit omniprezent pe PC-uri, precum și pe multe monitoare, proiectoare și televizoare de înaltă definiție.

A doua parte a proiectului presupune procesarea imaginii pe care dorim să o afisam, aplicându-i diverse filtre. Prelucrarea digitală a imaginilor constă în utilizarea unui computer digital pentru a procesa imagini digitale printr-un algoritm. În contextul actual, acesta este un domeniu de interes, fiind folosit în cadrul unor tehnologii de actualitate, precum computer vision, pattern recognition sau remote sensing.

Fie că suntem conștienți de ea sau nu, procesarea de imagini este peste tot în viața noastră de zi cu zi. În primul rând, fotografii filtrate sunt omniprezente în social media, articole de știri, reviste, cărți, peste tot. Dacă ne gândim la imagini ca funcții care mapează locațiile din imagini la valorile pixelilor, atunci filtrele sunt doar sisteme care formează o imagine nouă și, de preferință, îmbunătățită dintr-o combinație a valorilor pixelilor imaginii originale. Asadar, obiectivele proiectului sunt:

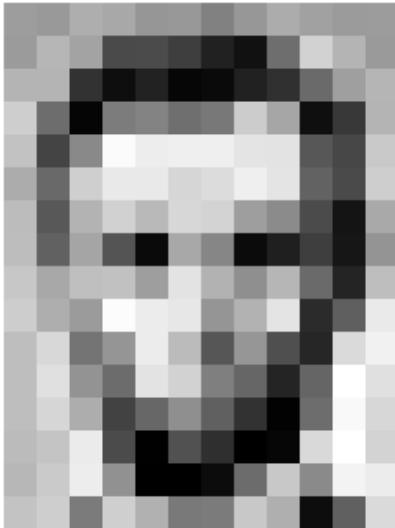
- Gasirea unei soluții de stocare a imaginii și un format al imaginii compatibil;
- Definirea unui driver ce corespunde protocolului de afisare prin interfața VGA și permite afisarea corectă pe ecranul pe care il avem la dispozitiv;
- Proiectarea unui automat de stari ce trece prin toate filtrele pe care dorim să le aplicăm;
- Definirea unui cod pentru aplicarea filtrului GreyScale;
- Definirea unui cod pentru aplicarea filtrului Edge Detection;

Chapter 3

Fundamentare teoretica

3.1 Reprezentarea imaginilor

O imagine este reprezentata in domeniul calculatoarelor ca o matrice de pixeli. Putem privi o imagine ca o functie, unde fiecarui pixel ii este asociata o valoare, ce reprezinta o culoare.



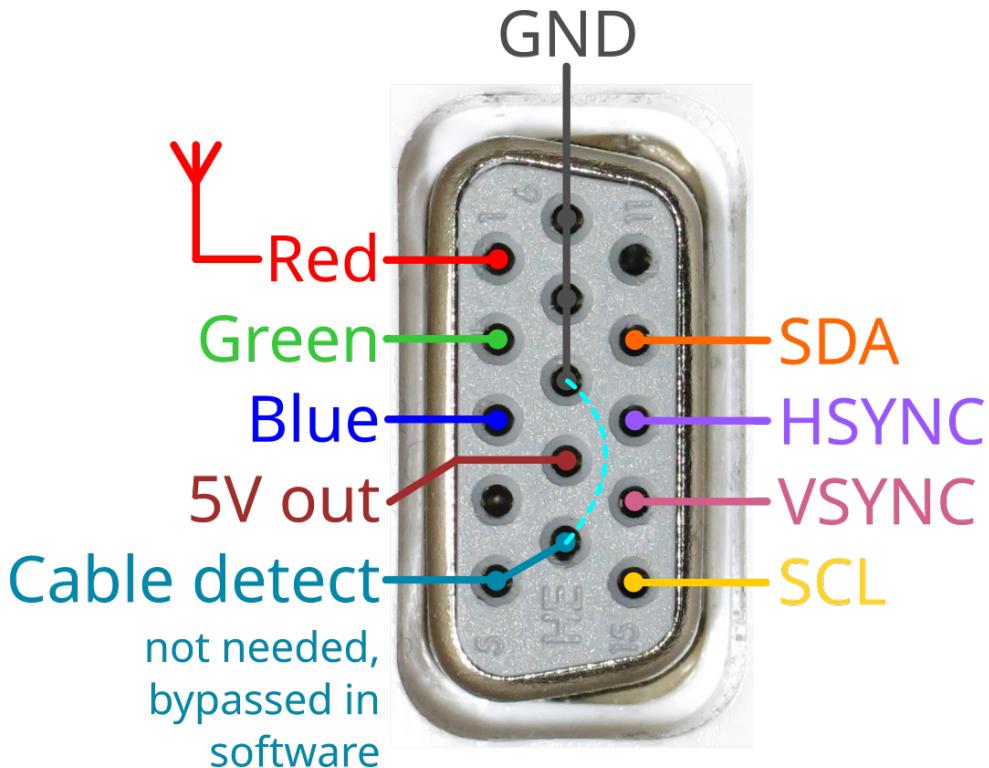
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	6	124	131	111	120	204	166	15	56	180
194	68	197	251	297	299	299	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	159	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	158	252	236	231	149	178	228	43	95	234
190	216	116	149	236	167	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	236	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	209	138	243	236
195	206	123	297	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	6	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	159	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	158	252	236	231	149	178	228	43	95	234
190	216	116	149	236	167	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	236	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	209	138	243	236
195	206	123	297	177	121	123	200	175	13	96	218

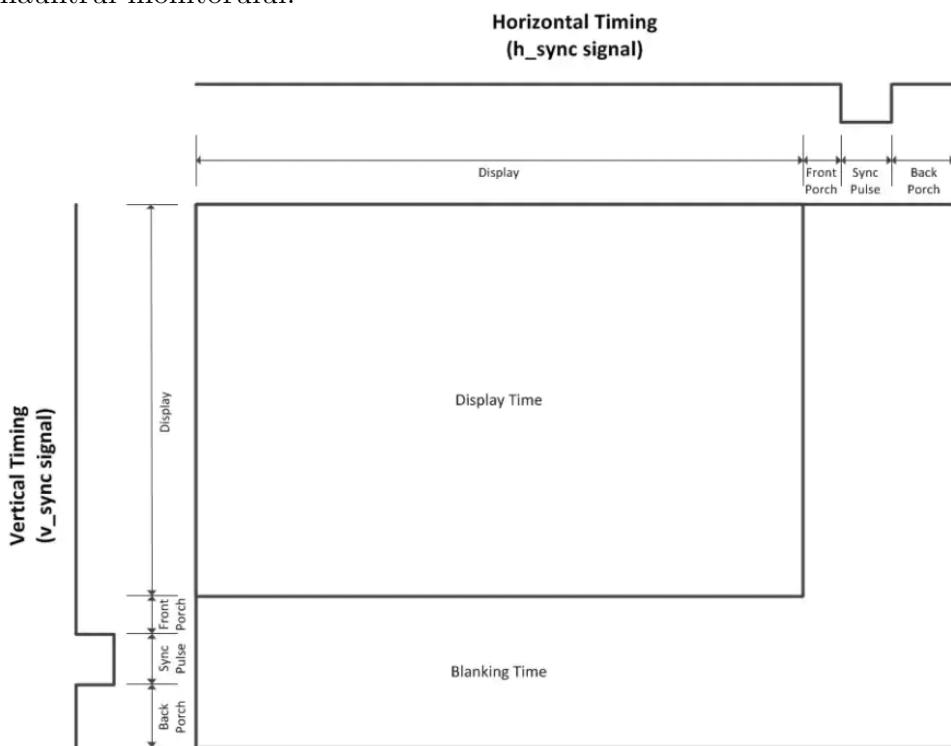
In sistemul de reprezentare RGB fiecare culoare este definita ca o combinatie dintre rosu, verde si galben, in diferite proportii. Asadar, fiecare culoare este definita printr-un vector de 3 valori de la 0 la 255, fiecare valoare reprezentand intensitatea fiecarui dintre canalele red, green yellow. Există un total de $256 \times 256 \times 256 = 16.777.216$ combinații sau opțiuni de culoare.

3.2 Interfata VGA

Un Video Graphics Array (VGA) este un conector standard folosit pentru video output catre monitoare. Conectorul are 15 pini, asezati in trei randuri. Acesti pini cara semnal video analog RGBHV: red, green, blue, horizontal sync, vertical sync.



Vertical si Horizontal Sync sunt semnale digitale de 0V/5V folosite sa sincronizeze transmiterea datelor cu monitorul. Fiind semnale digitale, acestea pot fi transmise direct de catre FPGA. Pe canalele RGB se transmit semnale analog intre 0V si 0.7V. Cele trei fire de culoare sunt terminate cu rezistente de 75ohm. De asemenea sunt terminate cu rezistente de 75ohm si inauntrul monitorului.



In Figura de mai sus ne sunt prezentate semnalele de timp produse de controllerul VGA. Controllerul contine 2 numaratoare. Unul se incrementeaza o data cu fiecare pixel si controleaza Horizontal sync. Daca numaratorul incepe la valoarea 0, numaratorul va afisa indexul pixelului in timpul afisarii. Timpul de afisare este urmat de un blanking time, ce are 3 caracteristici:

horizontal front porch, the horizontal sync pulse itself, si horizontal back porch, fiecare avand o durata specifică. La sfârșitul randului numaratorul se poate resetă. Cel de-al doilea numarator se va incrementa o dată cu completarea fiecarui rand, controlând semnalul de Vertical sync. La fel ca mai înainte, dacă counterul începe de la 0 acest număr va coincide cu randul pe care ne aflăm. Timpul de afisare este urmat de un blanking time cu corespondentele sale front porch, sync pulse, and back porch. După ce numaratorul vertical se resetează se va începe desenarea și afisarea următorului ecran.

Enableul ecranului este definit de și logic între counterele pentru semnalele orizontal și vertical.

Același cablu VGA poate fi folosit cu o varietate de rezoluții, de la 320×400 px @70 Hz, sau 320×480 px @60 Hz până la 1280×1024 px, toate cu timinguri specifice pentru display și blanking time.

3.3 Procesarea imaginilor

Există două tipuri de procesare a imaginilor:

- Image filtering: schimbă culorile pixelilor unei imagini, fără a le altera poziția.
- Image warping: schimbă domeniul imaginii, deci poziția pixelilor. Asadar, pixelii sunt mapati la poziții noi în cadrul imaginii, fără a le schimba valoarea (culoarea).

Scopul utilizării filtrelor este de a modifica sau îmbunătăți proprietățile imaginii și/sau de a extrage informații valoroase din imagini, ar fi muchii, colțuri și pete. În cadrul acestui proiect vom implementa filtre de tipul image filtering: greyscale, binary și edge detection.

3.4 Filtrul Greyscale

Filtrul greyscale își propune să convertească o imagine viu colorată în echivalentul său cu nuanțe de gri. În domeniul RGB, culoarea gri este reprezentată prin folosirea acelasi valori intregi pentru fiecare canal red, green, yellow. Magnitudinea acestei valori indică intensitatea nuanței de gri.

O soluție pentru convertirea unei imagini în format greyscale realizează media canalelor RGB pentru fiecare pixel și suprascrie fiecare canal cu aceasta valoare.

3.5 Filtrul Binary

Filtrul binary convertește o imagine doar în pixeli de culoare albă sau neagră. În format RGB, acest lucru înseamnă că fiecare pixel va avea ori valoarea (0,0,0)-negru sau (255,255,255)-alb. Acest filtru dă o reprezentare care va fi utilă pentru aplicarea altor filtre, precum edge detection.

$$g[n, m] = \begin{cases} 255, & f[n, m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$



O solutie pentru conversia unei imagini in binary converteste initial imaginea in format greyscale si apoi schimba valoarea inscrisa pe fiecare canal in 0 sau 255, in functie de threshhold-ul stabilit.

3.6 Filtrul Edge Detection

Filtrul edge detection converteste o imagine intr-o reprezentare binary care evidentaiza marginile figurilor din imagine.



O solutie pentru aplicarea filtrului edge detection respecta urmatorul algoritm:

- Imaginea se converteste in binary.
- Se extrage un pixel (de la pozitia xy) impreuna cu pixelii lui vecini, formandu-se o matrice de 3x3 pixeli.
- Se definesc mastile pentru detectia liniilor orizontale, respectiv verticale

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

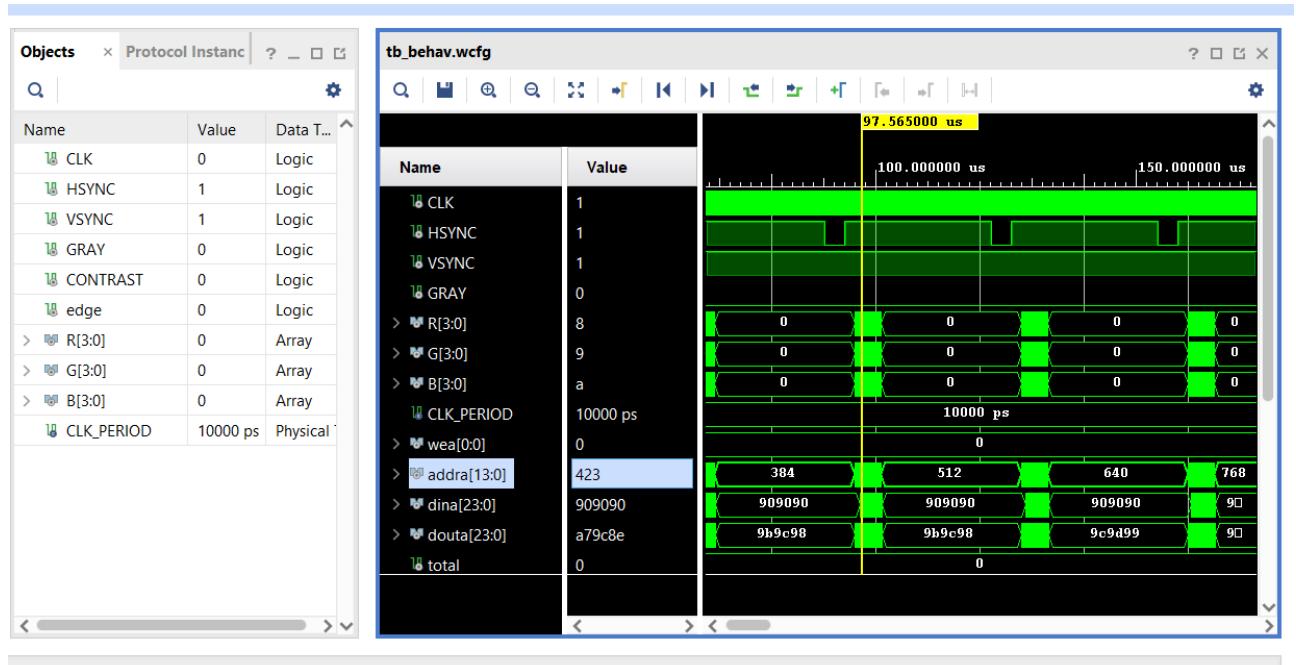
Gy

- Se aplica masca pentru detectia liniilor orizontale matricii de 3x3 aferente unui pixel: valoarea canelelor RGB de pe pozitia ij din matricea aferenta unui pixel se inmulteste cu valoarea din matricea masca aferenta pozitiei ij. Se realizeaza suma valorilor din matricea obtinuta si se retine (s1).
- Se aplica masca pentru detectia liniilor verticale matricii de 3x3 aferente unui pixel. Se realizeaza suma valorilor din matricea obtinuta si se retine (s2).
- Se realizeaza media s1 si s2 si, in functie de un threshold stabilit anterior, intr-o imagine noua se inscrie pixelul de culoare alba sau negra la pozitia xy.
- Algoritmul se repeta pentru fiecare pixel din matricea de start, iar la finalul parcurgerii acesteia, imaginea noua formată va contine rezultatul dorit. Se modifica thresholdul daca nivelul de sensibilitate nu este cel asteptat.

Chapter 4

Proiectare si implementare

Metoda experimentală utilizată pentru verificarea progresului este, în principal, mediul de simulare din mediul Vivado. Observarea și analizarea rezultatului afisării pe ecran s-a dovedit de asemenea facil.



4.1 Driver VGA

```
library IEEE;
use IEEE.STD.LOGIC_1164.ALL;
use IEEE.STD.LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity vga_driver is
    Port ( CLK : in STD_LOGIC; —100 MHz Basys3
           HSYNC : out STD_LOGIC;
           VSYNC : out STD_LOGIC;
```

```

        R,G,B : out STD_LOGIC_VECTOR (3 downto 0));
end vga_driver;

architecture Behavioral of vga_driver is

signal ClockDiv4: STD_LOGIC; — 25 MHz Clock

constant picture_edge : Integer:=128;
constant picture_size : Integer:=16348; — 128*128

— Signals for Block RAM
signal wea : STD_LOGIC_VECTOR(0 DOWNTO 0):="0";
signal addra : STD_LOGIC_VECTOR(13 DOWNTO 0):=(others=>'0');
signal dina : STD_LOGIC_VECTOR(23 DOWNTO 0):=(others=>'0');
signal douta : STD_LOGIC_VECTOR(23 DOWNTO 0):=(others=>'0');

constant HD : integer := 639; — Horizontal Display
constant HFP : integer := 16; — Horizontal front porch
constant HSP : integer := 96; — Horizontal Sync pulse
constant HBP : integer := 48; — Horizontal back porch

constant VD : integer := 479; — Vertical Display
constant VFP : integer := 10; — Vertical front porch
constant VSP : integer := 2; — Vertical Sync pulse
constant VBP : integer := 33; — Vertical back porch

signal hPos : integer := 0;
signal vPos : integer := 0;

signal videoOn : std_logic := '0';
signal clk50: std_logic := '0';

component pozica IS
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(23 DOWNTO 0)
  );
END component;

begin

  clk_div:process(CLK)
  begin
    if(CLK'event and CLK = '1')then
      clk50 <= not clk50;

```

```

end if;

if(clk50'event and clk50 = '1') then
    ClockDiv4 <= not ClockDiv4;
end if;
end process;

```

--RAM cu poza
U3: pozica Port map (clka=>ClockDiv4 , wea=>wea , addra=>addra , dina=>dina , douta=>douta);

--divizor de frecventa
Horizontal_position_counter:process(ClockDiv4)
begin
 if(ClockDiv4'event and ClockDiv4 = '1')then
 if (hPos = (HD + HFP + HSP + HBP)) then
 hPos <= 0;
 else
 hPos <= hPos + 1;
 end if;
 end if;
end process;

Vertical_position_counter:process(ClockDiv4 , hPos)
begin
 if(ClockDiv4'event and ClockDiv4 = '1')then
 if(hPos = (HD + HFP + HSP + HBP))then
 if (vPos = (VD + VFP + VSP + VBP)) then
 vPos <= 0;
 else
 vPos <= vPos + 1;
 end if;
 end if;
 end if;
end process;

Horizontal_Synchronisation:process(ClockDiv4 , hPos)
begin
 if(ClockDiv4'event and ClockDiv4 = '1')then
 if((hPos <= (HD + HFP)) OR (hPos > HD + HFP + HSP))
 then
 HSYNC <= '1';
 else
 HSYNC <= '0';
 end if;
 end if;
end process;

```

        end if;
    end if;
end process;

Vertical_Synchronisation:process(ClockDiv4 , vPos)
begin

    if(ClockDiv4'event and ClockDiv4 = '1')then
        if((vPos <= (VD + VFP)) OR (vPos > VD + VFP + VSP))
            then
                VSYNC <= '1';
        else
                VSYNC <= '0';
        end if;
    end if;
end process;

video_on:process(ClockDiv4 , hPos , vPos)
begin

    if(ClockDiv4'event and ClockDiv4 = '1')then
        if(hPos <= HD and vPos <= VD)then
            if(hpos <= picture_edge and vpos<=
                picture_edge) then
                videoOn <='1';
            else
                videoOn <='0';
            end if;
        else
                videoOn <= '0';
        end if;
    end if;
end process;

draw:process(ClockDiv4 , hPos , vPos , videoOn)
begin

    if(ClockDiv4'event and ClockDiv4 = '1')then
        if(videoOn = '1') then
            B<=douta(23 downto 20); G<=douta(15 downto
                12); R<=douta(7 downto 4);
            addra<=STD_LOGIC_VECTOR(unsigned(addr)+1);
        else
            R<=(others=>'0'); G<=(others=>'0'); B<=(others
                =>'0');
        end if;
    end if;

```

```

if ( unsigned( addra )>=picture_size -1) then
    addra<=(others=>'0');
end if;

end process ;

end Behavioral;

```

Mai intai am definit ca si constante valorile standard pentru timinguri, pentru a putea fi usor de folosit.

In continuare,in procesul clkdiv am generat clockul de 25MHz pornind de la clockul placutei de 100MHz. Desi rezolutia 640x480@60Hz presupune o frecventa de afisare a pixelilor de 25.175MHz am descoperit ca frecventa de 25MHz generata de divizorul simplu este suficient de buna.

In urmatoarele doua procese am generat in mod similar semnalele counterelor de pozitie: vertical si orizontal.

Urmardind figura pentru semnalele ce trebuie produse de controllerul FPGA, am generat apoi iesirile VSYNC si HSYNC.

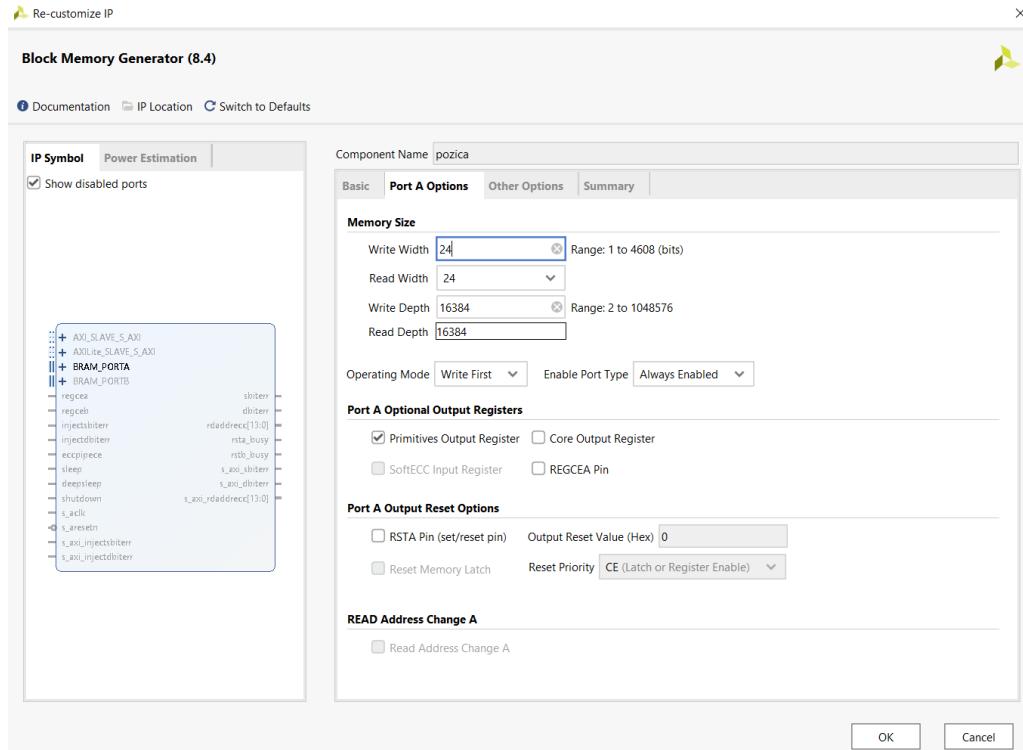
In procesul videoon am determinat semnalul VideoOn, care indica perioada in care trebuie afisati pixeli din imagine. Daca videoon este 0, pe canalul RGB se va transmite negru. Avem grija sa dezactivam semnalul cand iesim din range-ul fotografiei, nu numai din cel al ecranului.

Procesul draw determina ce este transmis pe canalul RGB in fiecare moment al programului, in functie de VideoOn. Daca VideoOn este 1, atunci este extras un pixel din blockul de memorie ram si afisat. Retinem ca un cuvant din ram are 24 de pixeli, deci fiecare canal de culoare are 8 biti. Deoarece canal de culoare prin VGA are doar 4 biti, alegem sa trimitem pe iesire doar primii 4 cei mai semnificativi biti ai fiecarei culori. Daca VideoOn este 0, atunci transmitem negru. Avem grija sa resetam cursorul (addra) ce parurge fotografia din memorie pixel cu pixel cand ajungem la finalul imaginii de afisat.

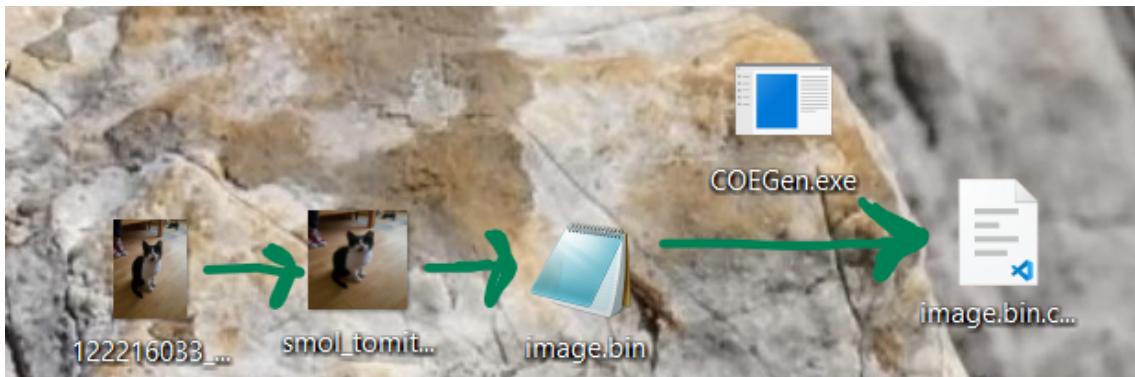
4.2 Stocarea imaginii

Deoarece imagea pe care vrem sa o afisam pe ecran are dimensiuni foarte mari, am ales sa o stocam pe un block de memorie RAM integrate in FPGA.

<https://www.nandland.com/articles/block-ram-in-fpga.html>

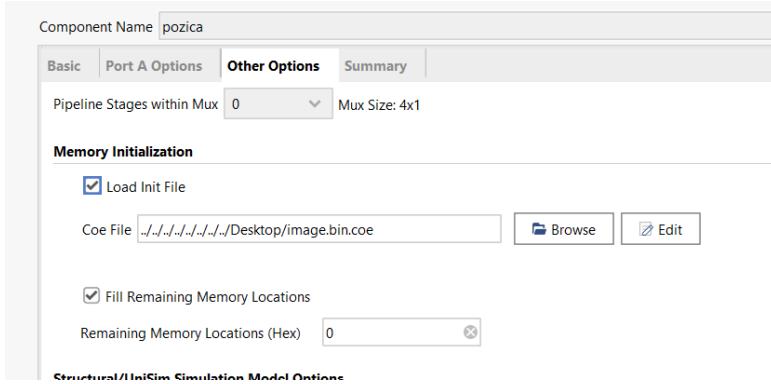


Blocul RAM va stoca o poza de dimensiune 128X128. Fiecare cuvant reprezinta un pixel si retine culoarea lui reala (true RGB) pe 24 de biti. Astfel, memoria va avea 128*128 de adrese la care se afla cate 24 de biti.



Pentru a inscrie o imagine in memoria block ram am respectat urmatorii pasi:

1. am redimensionat fotografia dorita, ca sa fie un patrat (pentru simplitate) de dimensiune 128x128. Am folosit o poza atat de mica fiindca nu avem mult spatiu in blockul ram.
2. am convertit imaginea in binar
3. am folosit COEGen.exe pentru a genera fisierul .coe care contine formatul dorit (true RGB pentru fiecare pixel).
4. am incarcat imaginea in blockul ram;



A fost nevoie de un block de memorie aditional pentru stocarea imaginii pe care s-a aplicat filtrul edge detection. Aceasta memorie nu a fost initializata.

Blockurile de memorie ram au fost instantiatate in codul VHDL in felul urmator si le vom folosi ca pe niste memorii obisnuite, unde wea este semnalul de enable pt scriere, addra este adresa de intrare, dina sunt datele de intrare si douta este semnalul de iesire:

```

component pozica IS
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(23 DOWNTO 0)
  );
END component ;

component poza_proc IS
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(23 DOWNTO 0)
  );
END component ;

azteca: poza_proc port map( clka=>ClockDiv4 , wea=>wea_proc , addra=>
  addra , dina=>dina_proc , douta=>douta_proc );
U3: pozica Port map ( clka=>ClockDiv4 , wea=>wea , addra=>addra , dina=>
  dina , douta=>douta );

```

4.3 Filtrul greyscale si binary

```

grayscale:process(ClockDiv4 , mode , douta)
begin

```

```

if(ClockDiv4'event and ClockDiv4 = '1')then
    if(mode = '0') then
        dina(23 downto 20)<= std_logic_vector(to_unsigned((
            to_integer(unsigned(douta(23 downto 20))) + to_integer(
            unsigned(douta(15 downto 12))) + to_integer(unsigned(
            douta(7 downto 4)))) / 3 , dina(23 downto 20)'length)) ;
        dina(15 downto 12)<= std_logic_vector(to_unsigned((
            to_integer(unsigned(douta(23 downto 20))) + to_integer(
            unsigned(douta(15 downto 12))) + to_integer(unsigned(
            douta(7 downto 4)))) / 3 , dina(23 downto 20)'length)) ;
        dina(7 downto 4)<= std_logic_vector(to_unsigned((to_integer(
            unsigned(douta(23 downto 20))) + to_integer(unsigned(
            douta(15 downto 12))) + to_integer(unsigned(douta(7
            downto 4)))) / 3 , dina(23 downto 20)'length)) ;
    elsif(mode = '1') then
        if(douta(23 downto 20)> "0111") then
            dina(23 downto 20)<= "1111";
        else
            dina(23 downto 20)<="0000";
        end if ;

        if(douta(15 downto 12)> "0111") then
            dina(15 downto 12)<= "1111";
        else
            dina(15 downto 12)<="0000";
        end if ;

        if(douta(7 downto 4)> "0111") then
            dina(7 downto 4)<= "1111";
        else
            dina(7 downto 4)<="0000";
        end if ;
        end if ;
        end if ;
    end process ;

```

Procesul greyscale se ocupa de convertirea imaginii in format greyscale sau binary in functie de semnalul mode. Cand mode este 0 se realizeaza greyscale: blockul de memorie ce contine poza este suprascris cu valorile greyscale corespunzatoare fiecarui pixel, dupa cum a fost prezentat algoritmul din sectiunea anterioara.

Automatul de stari ne trece apoi in starea in care mode este 1, care realizeaza conversia imaginii greyscale in binary. Avand imaginea greyscale in memorie, comparam valoarea de pe canalele RGB ale fiecarui pixel cu B0111. Daca ea este mai mare, suprascriem pixelul cu valoarea RGB pentru alb, iar daca este mai mica inscriem negru.

4.4 Filtrul edge detection

```
edge_detection: process(stare_edge, ClockDiv4, edgedetection)
```

```

begin
if ClockDiv4'event and ClockDiv4 = '1' then
  case stare_edge is
    when init =>
      if(edgedetection = '1')then
        stare_edge<=s1;
      else
        stare_edge<=init;
      end if;
      when s1 =>
        aux<=aux-PICTURE_WIDTH - 1;
        sumay <= sumay + (-1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        sumax <= sumax + (-1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        — de_adunat =
        stare_edge<= s2;
      when s2 =>
        aux<=aux+1;
        sumax <= sumax + (-2 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        — de_adunat =
        stare_edge<= s3;
      when s3 =>
        aux<=aux+1;
        sumay <= sumay + (1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        sumax <= sumax + (-1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        — de_adunat =
        stare_edge<= s4;
      when s4 =>
        aux<=aux + PICTURE_WIDTH * 2 -2;
        sumay <= sumay + (-1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        sumax <= sumax + (1 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        — de_adunat =
        stare_edge<= s5;
      when s5 =>
        aux<=aux + 1;
        sumax <= sumax + (2 *(to_integer(unsigned(douta(23
          downto 20))))) ;
        — de_adunat =
        stare_edge<= s6;

```

```

when s6 =>
    aux<=aux + 1;
    sumay <= sumay + (1 *(to_integer(unsigned(douta(23
        downto 20))))) ;
    sumax <= sumax + (1 *(to_integer(unsigned(douta(23
        downto 20))))) ;
    — de_adunat =
    stare_edge<= s7;

when s7 =>
    aux<=aux - PICTUREWIDTH -2;
    sumay <= sumay + (-2 *(to_integer(unsigned(douta(23
        downto 20))))) ;
    — de_adunat =
    stare_edge<= s8;
when s8 =>
    aux<=aux + 2;
    sumay <= sumay + (2 *(to_integer(unsigned(douta(23
        downto 20))))) ;
    — de_adunat =
    stare_edge<= s9;
when s9 =>
    aux<=aux-1;
    sumax<=(sumax+sumay) /2;
    if (sumax>10)then
        dina_proc<="111111111111111111111111";
    else
        dina_proc<="00000000000000000000000000000000";
    end if;
    stare_edge<= s10;
when s10 =>
    sumax<=0;
    sumay<=0;
    aux<=aux+1;
    if (aux = picture_size -1 )then
        stare_edge <= gata;
    else
        stare_edge<=s1;
    end if;
when gata =>
    stare_edge <= gata;

when others =>
end case;

end if;

end process;

```

```

alege : process (stare_edge)
begin

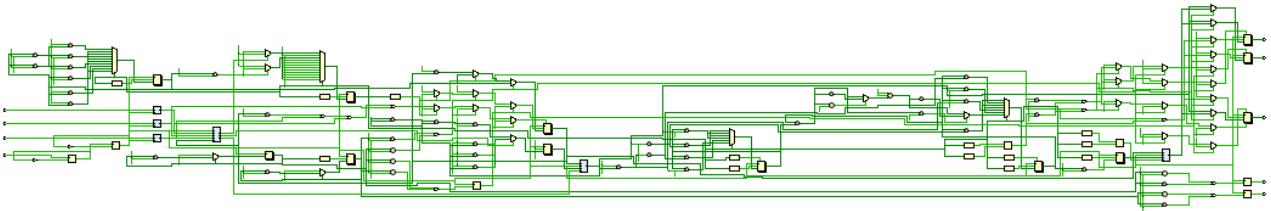
case stare_edge is
when init => afisare <='1'; wea_proc<="0" ; mem2<='0';
when s1=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s2=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s3=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s4=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s5=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s6=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s7=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s8=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when s9=>afisare <='0'; wea_proc <="1" ; mem2<='0';
when s10=>afisare <='0'; wea_proc <="0" ; mem2<='0';
when gata=>afisare <='1'; wea_proc <="0" ; mem2<='1';
when others => afisare <='0';

```

Implementarea noastră pentru convertirea unei imagini pentru detectarea marginilor urmează algoritmul descris în secțiunea anterioară. Astfel, după realizarea imaginii în greyscale și apoi binary, automatul de stări ne va trece în starea edgedetection, care porneste algoritmul de creare a noii imagini. Procesul conține un automat de stări intern, definit de semnalul stare-edge. Stările posibile sunt următoarele:

- init: start calcul imagine procesată
- s1-s8: deplasarea pe poziția pixelului vecin dorit și actualizarea sumelor;
- s9: înscriverea valorii pixelului nou în a două memorie
- s10: deplasarea la urmatorul pixel din imaginea de procesat și actualizarea sumelor sx și sy;
- gata: finalizarea procesării imaginii: imaginea din a două memorie este afisată pe ecran;

4.5 Diagrama RTL



4.6 Manual de utilizare

Pentru testarea adecvata a functionalitatilor proiectului se respecta urmatoarele instructiuni in ordine:

- Se alimenteaza monitorul si placuta;
- Se conecteaza placuta la monitor printr-un cablu VGA;
- Pe ecran apare iamginea inscrisa in blockul de memorie 1;
- Se apasa pe butonul din mijloc al placutei pentru a afisa imaginea in greyscale;
- se apasa pe butonul de sus al placutei pentru a afisa pe ecran imaginea in binary;
- se apasa pe butonul din dreapta al placutei pentru a afisa imaginea cu filtrul de edge detection;

Chapter 5

Rezultate experimentale

5.1 Testare

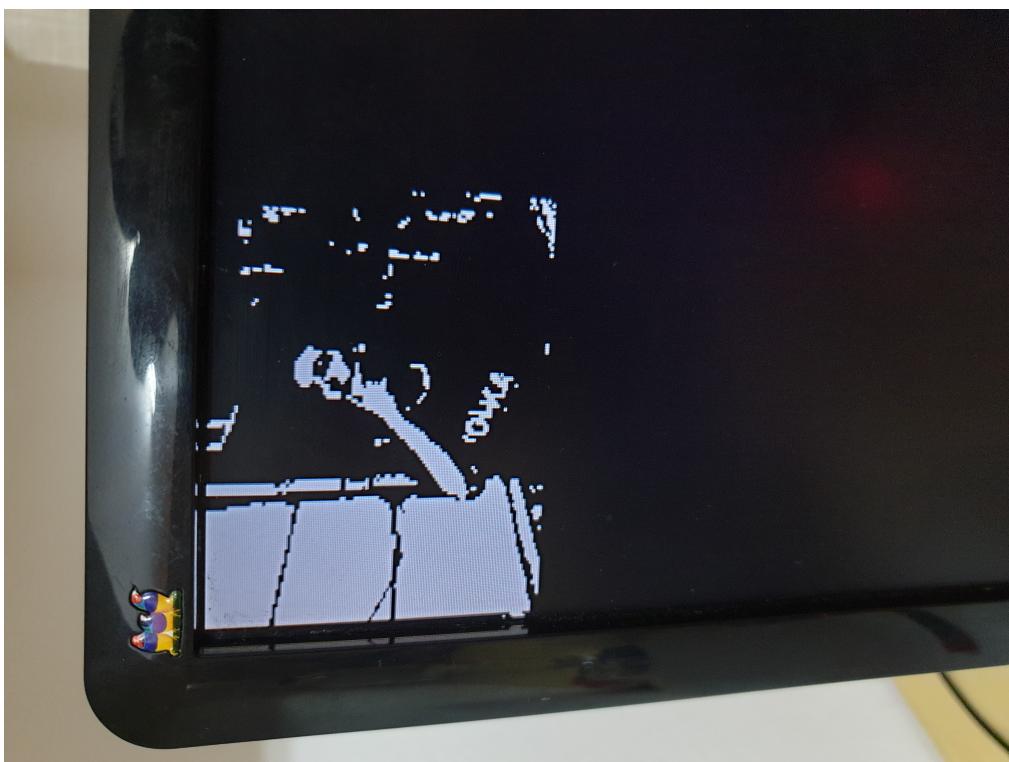
Realizand conexiunile din manualul de utilizare, aceasta este imaginea ce sa afiseaza initial. Imaginea se afiseaza corect, si arata ca in fisierul incarcat, cu exceptia unor pixeli care nu au aceeasi culoare fiindca am folosit doar cei ami semnificativi biti din culoarea pixelilor.



Dupa apasarea butonului din mijloc, imaginea este convertita corect in greyscale, punctandu-se distinge aceleasi trasaturi ale imaginii initiale.



Dupa apasarea butonului sus, imaginea este convertita corect in binary, ea fiind nedeformata si continand numai pixeli albi si negri. Se asemenea, inca se pot distinge trasaturile imaginii initiale.



Dupa apasarea butonului din dreapta, imaginii ii este aplicat filtrul de edge detection in mod corect: imaginea este in binary, insa cu negru sunt evidențiate numai marginile figurilor din poza. Inca se poate distinge continutul imaginii.



5.2 Dificultati intalnite

Au existat dificultati mai ales la afisarea imaginii nemodificate pe ecran, incepand de la calculul semnalului de sincronizare:



De asemenea, am avut probleme la parcugerea in intregime a imaginii. Deoarece nu calculez bine capetele imaginii, cand o afisam aceasta se misca si se decala cu cativa pixeli.

Am avut probleme si la calculul imaginii cu filtru de detectie a marginilor:



Fiindca, spre deosebire de celelalte doua filtre, calcului unui pixel din noua imagine se realizeaza pe mai multe cicluri de ceas. Solutia gasita a fost realizarea unui nou automat de stari, special pentru calcului acestei imagini.

Chapter 6

Concluzii

In concluzie, proiectul nostru rezolva cu succes problema afisarii unei imagini prin portul VGA si procesarea acesteia folosind filtrele greyscale, binary si edgedetection.

Desi am folosit algoritmi consacrați pentru afisarea imaginilor prin VGA si procesarea imaginilor, implementarea noastră este una originală, pornind de la automatele de stări ce facilitează trecerea eficientă prin toate filtrele de image processing.

Aplicatia pentru transmiterea datelor prin conectorul VGA este utilă în dezvoltarea oricărui proiect care folosește aceată interfață. Pe de altă parte, există multe domenii care folosesc procesarea de imagini. Aplicatia noastră poate fi utilă în proiecte ce presupun computer vision, patternrecognition sau remote sensing.

Ulterior, proiectul poate fi dezvoltat adăugând mai multe filtre de image processing. Pe parte de VGA, am putea adăuga funcționalități pentru afisare pe ecrane de diferite rezoluții. Imaginile ar putea fi stocate, de exemplu, pe un modul Pmod SdCard, care ar permite lucrul cu imaginii de dimensiuni mai mari, blocul RAM de pe placuta având dimensiuni limitate.

Bibliography

<https://ai.stanford.edu/~syueung/cvweb/tutorial1.html>

<https://www.youtube.com/watch?v=l7rce6IQDWs>

<https://www.youtube.com/watch?v=uqY3FMuMuRo>

[channel = GreatScott](https://www.youtube.com/watch?v=ZNunxg7o8l0ab) [fbclid = IwAR2zgh9LlptNG](#)