

**Materia: Proiectarea Sistemelor Numerice**

**VGA Controller**

Proiect realizat de:

Zaharie Andrei si Jiroveanu Stefan

Grupa 30219 Anul I

Cuprins

[**1.** **Cerinta problemei** 3](#_Toc72142959)

[***1.1.*** Prezentare schema bloc 3](#_Toc72142960)

[***1.2.*** Componente***:*** 4](#_Toc72142961)

[***1.3.*** Semnificaţia notaţiilor efectuate şi a pinilor interfeţei cu exteriorul 9](#_Toc72142962)

[***1.4.*** Algoritmi generare imagine: 9](#_Toc72142963)

[2. **Implementare Logisim** 11](#_Toc72142964)

[**2.1.** Prezentare generale proiect**:** 11](#_Toc72142965)

[**2.2.** Prezentare Componente**:** 12](#_Toc72142966)

[**2.3.** Prezentare Moduri și Algoritmi de afișare**:** 13](#_Toc72142967)

[**2.4.** Modul Paint**.** 17](#_Toc72142968)

[**2.5.** Modul Abstract**.** 18](#_Toc72142969)

[3. **Justificarea solutiei alese** 20](#_Toc72142970)

[**4.** **Intructiuni de utilizare si intretinere** 20](#_Toc72142971)

[4.1. Intructiuni de selectare/ colorare imagine (MODE = 1): 21](#_Toc72142972)

[4.2. Intructiuni de Miscare imagine(Mode = 0): 21](#_Toc72142973)

[4.3. Instructiuni de Free Paint(MODE = 1 and DrawMode = 1): 22](#_Toc72142974)

[4.4. Intructiuni de Abstract Mode: 22](#_Toc72142975)

[4.5. Alte instructiuni generale: 23](#_Toc72142976)

[5. **Posibilitati de dezvoltare ulterioara** 23](#_Toc72142977)

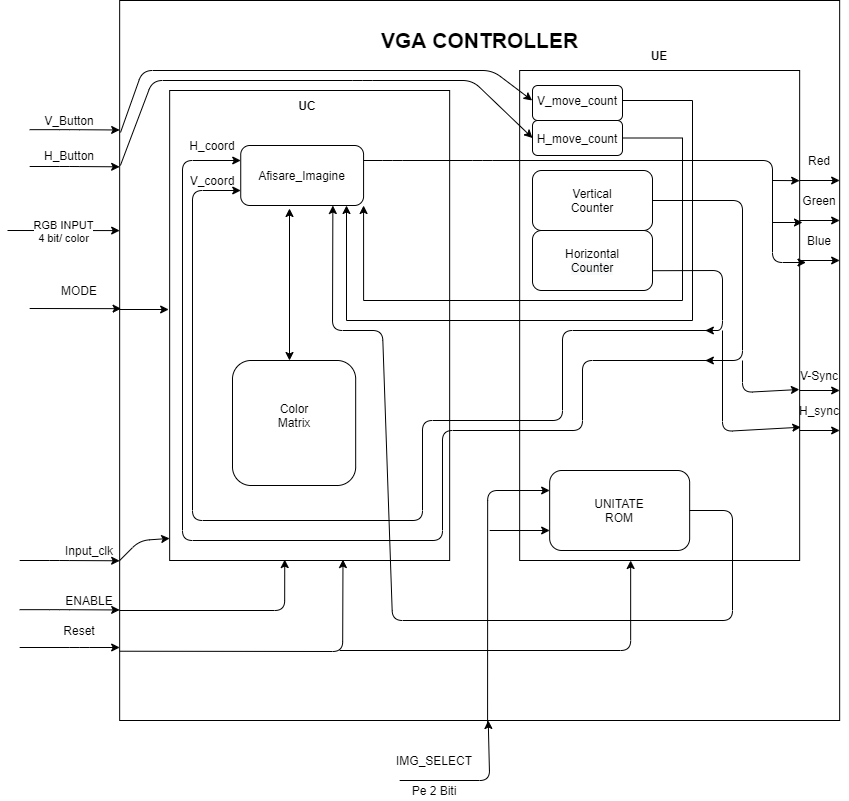
[6. **Distributie sarcini**: 26](#_Toc72142978)

[7. **Bibliografie**: 26](#_Toc72142979)

# **Cerinta problemei**

   Folosind plăcuţe cu FPGA să se implementeze un controller VGA. Trebuie afişate 4 imagini diferite, selecţia fiind citită de la butoanele plăcii. Imaginile trebuie să demonstreze abilitatea de selecţie a culorilor (minim 4 culori diferite). De asemenea, poziţionarea imaginilor pe ecran trebuie să fie controlabilă pe două axe cu ajutorul butoanelor. Sugestii de imagini: pătrat, dungi verticale, dungi orizontale, triunghi, cerc etc. Timpii necesari diferitelor rezoluţii se pot găsi, de exemplu, în tabelul 2-6 din documentația plăcii cu FPGA XUPV2P \_User\_Guide.pdf, la pag. 37-38. Diagramele de funcţionare şi explicaţii se găsesc în manualele de referinţă ale plăcilor cu FPGA.

* 1. Prezentare schema bloc:



## Componente**:**

* *Prezentarea componentelor***:**

1.  Color Matrix – matricea ce contine culorile fiecărui pixel din matricea de coordinate.

2.  Unitate ROM – contine codurile pentru fiecare din cele 4 imagini.

3.  Vertical + Horizontal Counter: 2 numaratoare care afiseaza toti pixeli ecranului pe rand (linie si coloana);

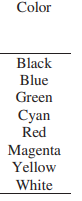
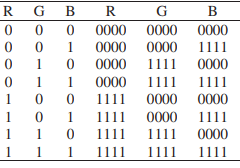
4. Afisare\_img.

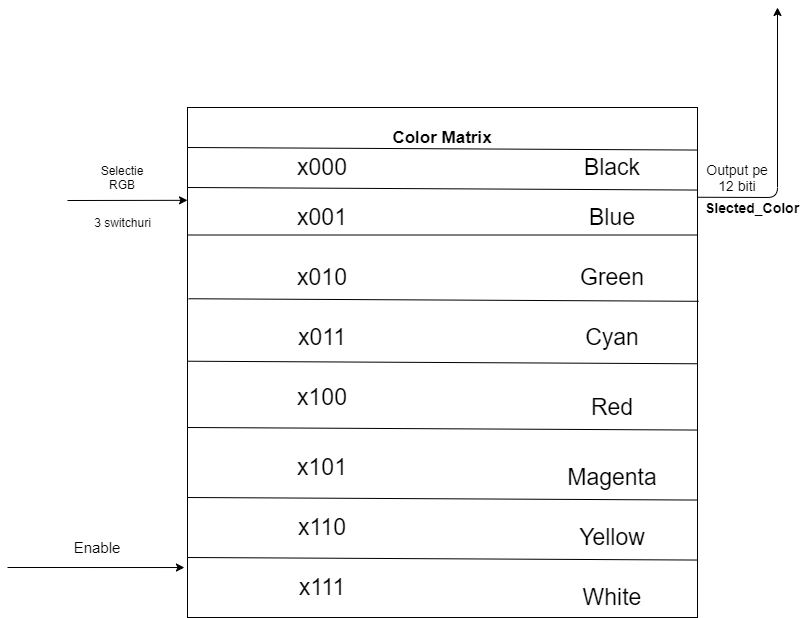
5.  Unitatea de control (contine Color si Coordinate matrix)

6. Unitatea de execuție (contine counterele, Img\_ROM, V\_count, H\_count)

* *Descrierea Componentelor***:**

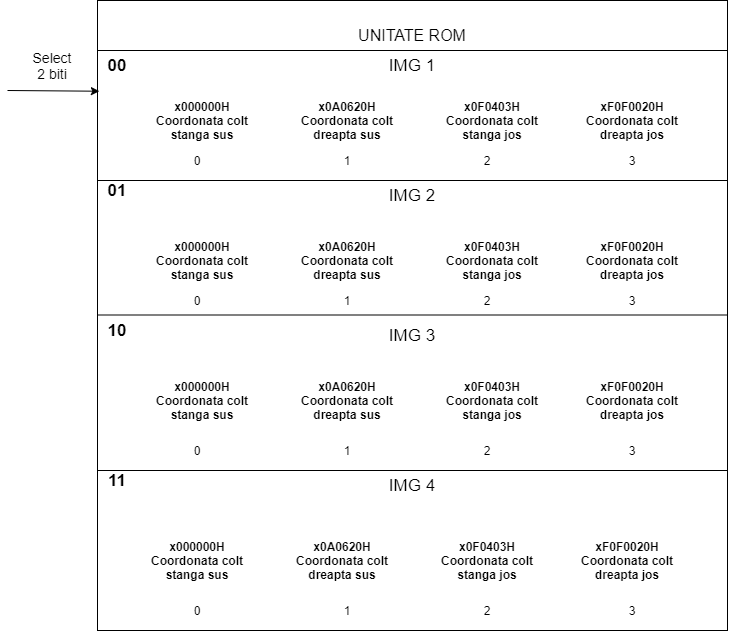
1. **Color Matrix** -  este o memorie de tip ROM ce este definită de tip **array(0 to 7) of bit\_vector (11 down to 0)** care defineste culorile pe care le avem astfel :



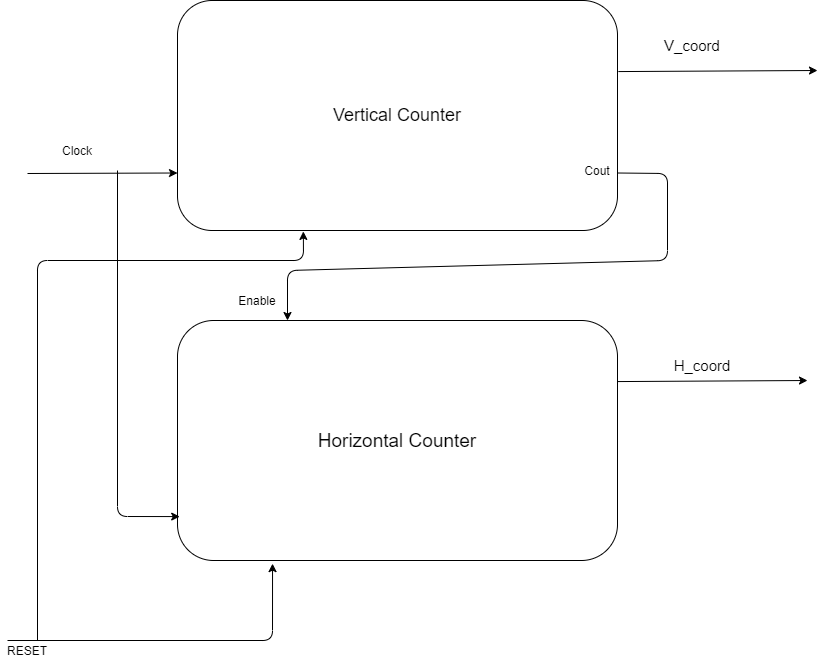


2.Unitatea ROM care va contine întreaga informație despre cele 4 poze pe care le putem afișa pe ecran.

Pentru cerc și triunghi vom folosi anumite formule matematice pentru a le genera.

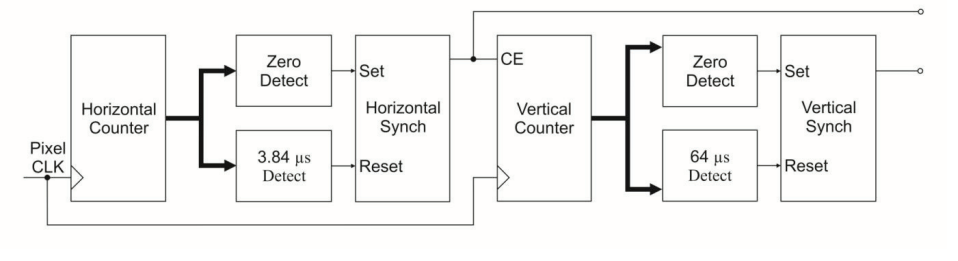


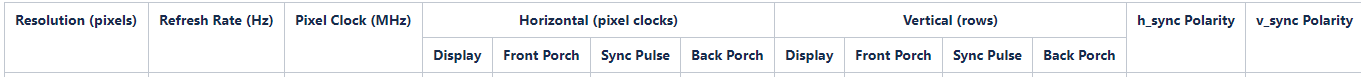
3. Vertical + Horizontal Counter sunt 2 countere cascade astfel incat horizontal counter sa creasca cu o unitate la fiecare ciclu terminat al vertical counterului.



Sistemul de sincronizare pentru setarea rezolutiei.

      Specificatii pentru rezoluția aleasă:

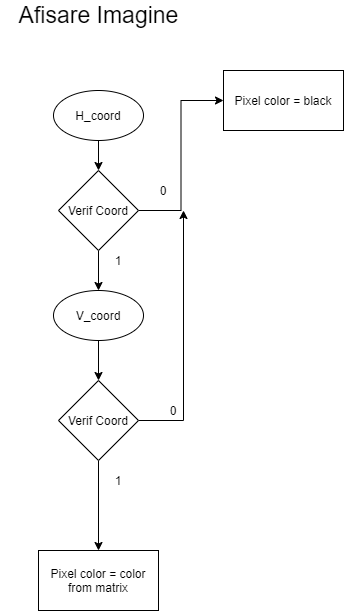


https://lh6.googleusercontent.com/7q-mJ7TETakMHBvVR1VukNO0oeCoVshtL8gXSGOuzi4ka3Q76-jDY8AXCNMti4QirA_DQIDKYIRtPXOEOLwzr-mJVpFR4kTnhcZxWaDKTEg9eYBUsm3dCQn9wkPmTjSt43YrDdCU

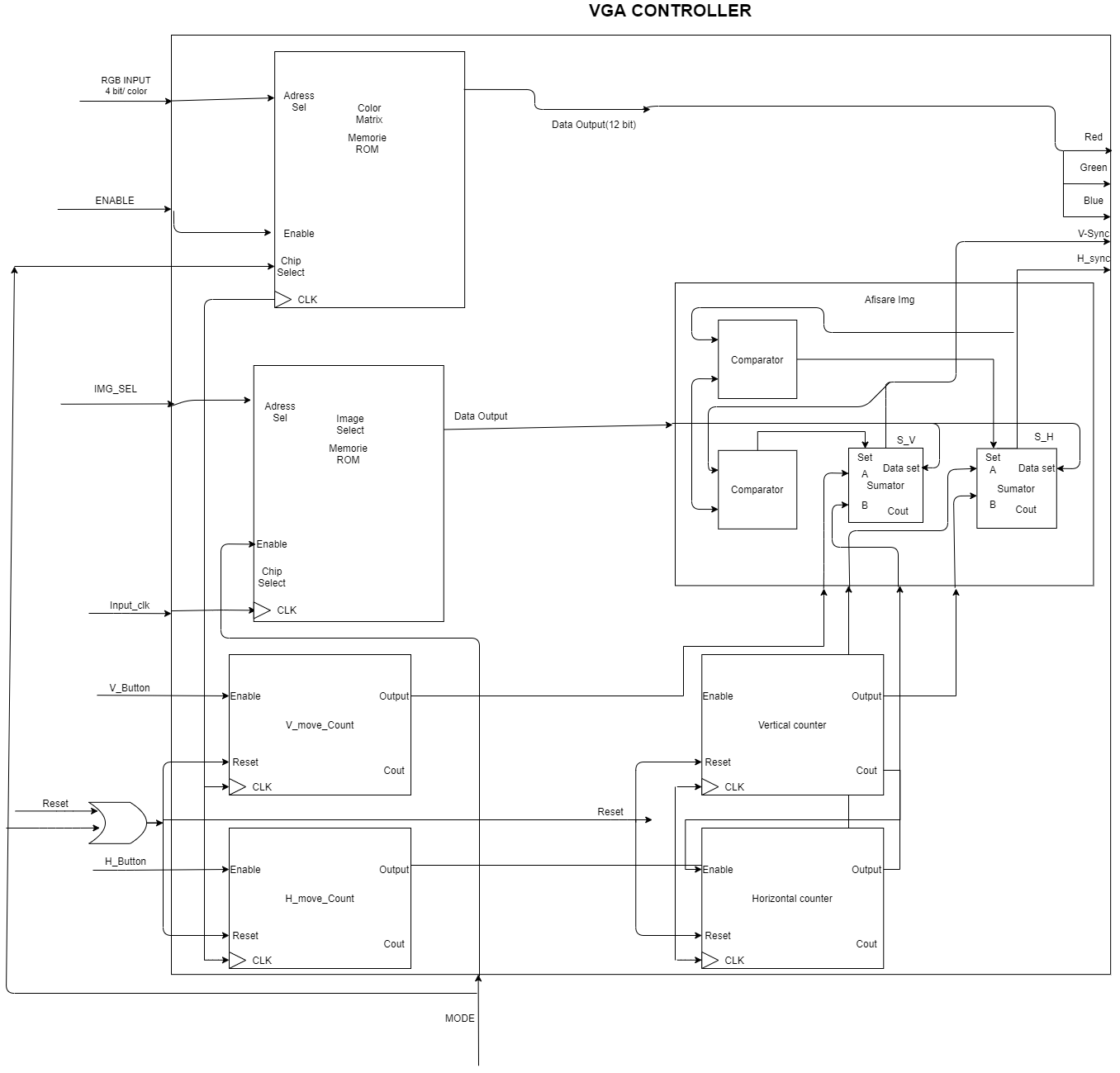
4. Afisare Imagine este o componenta care se ocupa cu verificarea daca imaginea este desenata corect in intervalul transmis de care unitatea ROM ce contine imaginile.

Daca coordonatele transmise de countere sunt validate atunci din ColorMatrix se va prelua culoarea cu care trebuie colorat pixelul de la coordonatele respective. In caz contrat se va colora cu negru care reprezinta background color.

Organigrama Afisare Imagine.



5. Unitate de Control - este responsabilă cu controlul execuției FPGA și are următorul algoritm descris de organigrama de mai jos.  
6.Unitatea de execuție se ocupă cu afișarea de imagini, controlul imaginii pe ecran și sincronizarea timpilor de afișare pentru a creea rezoluția dorită și este descrisă de următorul flow chart.

Schema extinsta controller vga

* 1. Semnificaţia notaţiilor efectuate şi a pinilor interfeţei cu exteriorul.

1. Inputuri :

* **V\_Button** - **input button** pentru controlul imagini pe axa OY și este pe 1 bit.
* **H\_Button** - **input button** pentru controlul imaginii pe axa OX și este pe 1 bit.
* **Dir\_Change** - **input switch** pentru schimbarea sensului de mers pentru butoanele H\_Button si V\_Button ( Jos & Dreapta cand Dir\_Change = 0 și Sus & Stanga cand Dir\_Change =1).
* **RGB\_INPUT** – 12 biti cate 4 biti pentru fiecare culoare(RGB). Vom folosi o memorie rom si **4 switchuri** pentru adrese.
* **Mode** – **input switch** pe un bit care selectează modul controlerului.
* **IMG\_Select** – **2 input switch** ( selecție pe 2 biti) care selectează una din cele 4 imagini.
* **Confirm\_Img** - **input button** care confirma imaginea aleasă pentru a fi tiparita pe ecran.
* **Input\_CLK** – clk pentru UC.
* **Enable** – **input switch** pe 1 bit pentru on/ off al controlerului.
* **Reset** –**input button** pe 1 bit pentru care reseteaza ecranul
* Total Switchuri folosite: 9.
* Total Butoane folosite : 4.

1. Outputuri :

* **Red** – pe 8 biti care transmite pe ecran ce culoare are pixelul.
* **Green** - pe 8 biti care transmite pe ecran ce culoare are pixelul.
* **Blue** - pe 8 biti care transmite pe ecran ce culoare are pixelul.
* **V\_Sync** – pe 1 bit care transmite coordonata lui OY a pixelului.
* **H\_Sync** – pe 1 bit care transmite coordonata pe OX a pixelului
* **LCD\_Enable** - pe 1 bit care controlează cand ecranul este activ pentru a se putea scrie imaginea pe el și cand nu.

## Algoritmi generare imagine:

1. **Algorithm generare cerc:**

for(int vPos=-radius; vPos<=radius; vPos++)// vpos ->[-radius; radius]

     for(int hPos=-radius; hPos<=radius; hPos++)// hpos ->[

        if(vPos\*vPos+hPos\*hPos <= radius\*radius)

            setpixel(origin.x+vPos, origin.y+hPos);

**Explicatie:**

Pentru generarea unui cerc folosim formula matematica prezentata mai sus. Aceasta verifica daca coordonatele generate de counterele noastre se afla in perimetrul unui cerc cu raza **“radius”.** Daca pixelul generat se afla in acest perimetru va fi colorat, in caz contrar va ramane culoarea backgroundului.

1. **Algorithm generare triunghi dreptunghic:**

for(int vPos=Apos; vPos<=Bpos; vPos++)// vpos ->[-radius; radius]

     for(int hPos=Cpos; hPos<=Dpos; hPos++)// hpos ->[

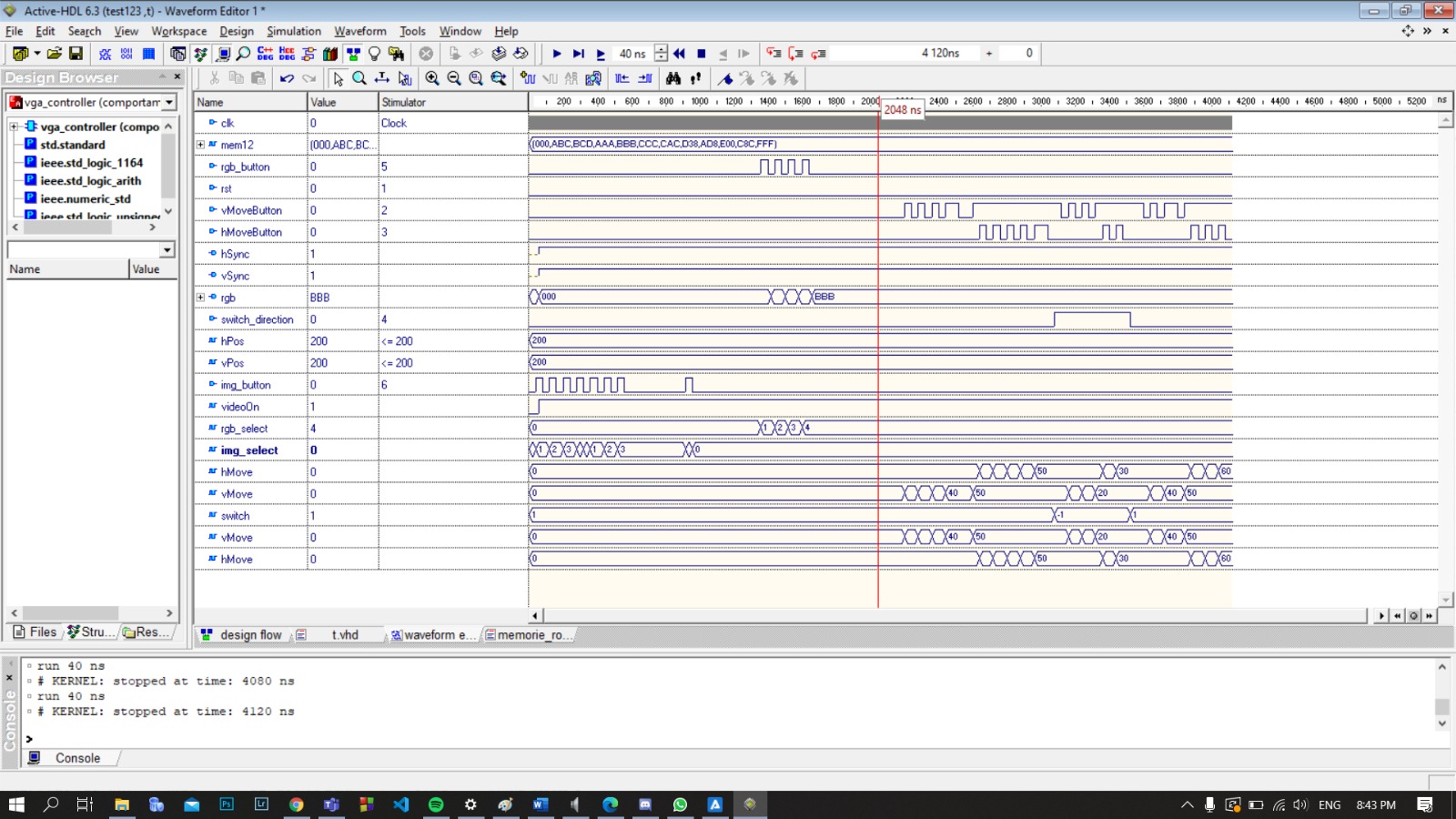
        if (vPos + hMove <= hPos + vMove) then

rgb <= mem(rgb\_select);

**Explicatie:**

Pentru generarea unui triunghi dreptunghic folosim formula de verificare a tuturor punctelor de sub diagonal principal a unui patrat.Astfel algoritmul genereaza un patrat dar coloreaza doar jumate din acesta urmand ca cealalta jumatate sa fie colorata in cluoarea de background.

Simulare VHDL:



# 2. **Implementare Logisim**

Proiectul creat in Logisim are aceeași structură ca și programul din VHDL, la care i-am adaugat inca cateva functionalitati în plus față de cerința proiectului.  
 **Functionalitati necesare**:

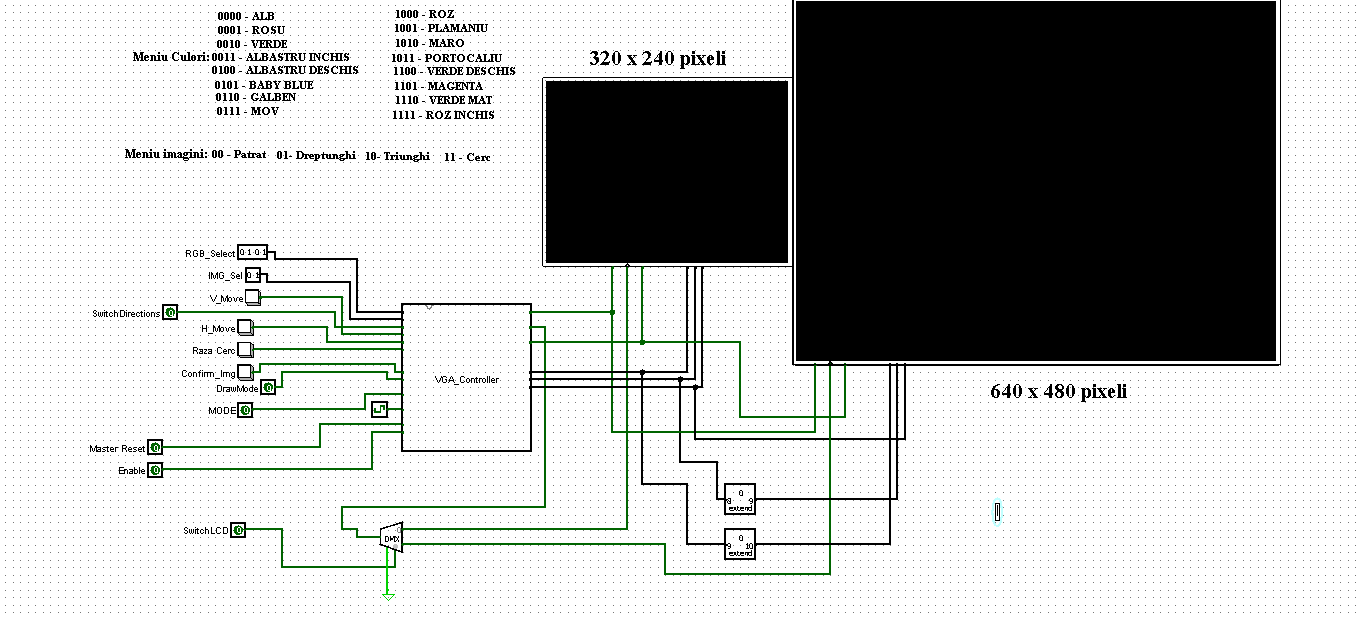
1. Afisare 4 tipuri de imagini (cerc, triunghi, dreptunghi, pătrat).
2. Miscare imagine (orizontal si vertical).
3. Colorare imagine in minim 4 culori diferite.

**Functionalitati adaugate in plus:**

1. Crearea de forme abstracte(Generarea aleatorie de cercuri și forme)
2. Modul paint(Permite utilizatorului sa coloreze pe ecran)
3. 16 culori diferite cu posibilitate de extindere pana la 16,777,216 culori.

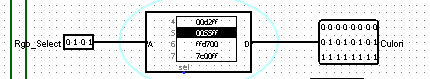
## **2.1.** Prezentare generale proiect**:**

In imaginea de mai sus se observa 2 ecrane de rezolutii diferite. In prezentare proiectului vom folosi ecranul cu rezolutia de 320x 240 pentru a accelera procesul de afișare al imaginii pe cat posibil.Din cauza faptului ca Logisim nu accepta un clock rate mai mare de 4.1 KHz. Iar pentru o funcționare optimă a programului noi avem nevoie de minim 25 MHz. Astfel pe un ecran cu o rezolutie mare precum cel de 640 x 480 printarea și colorarea imaginii ar dura prea mult ceea ce poate deveni deranjant.

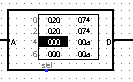


## **2.2.** Prezentare Componente**:**

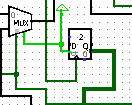
1. **Matrix Color**: Este o memorie Rom ce contine culorile disponibile.



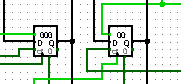
1. **Unitate ROM Imagini**: Este o memorie ROM care conține coordonatele imaginilor pe care dorim sa le afisam.



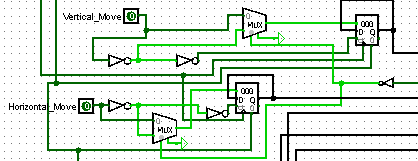
1. Clock Divider: Genereala semnalul de clock necesar pentru afișare.



1. Horizontal si Vertical Counter: Counterele care genereaza coordonatele pixelilor ce trebuie colorari.

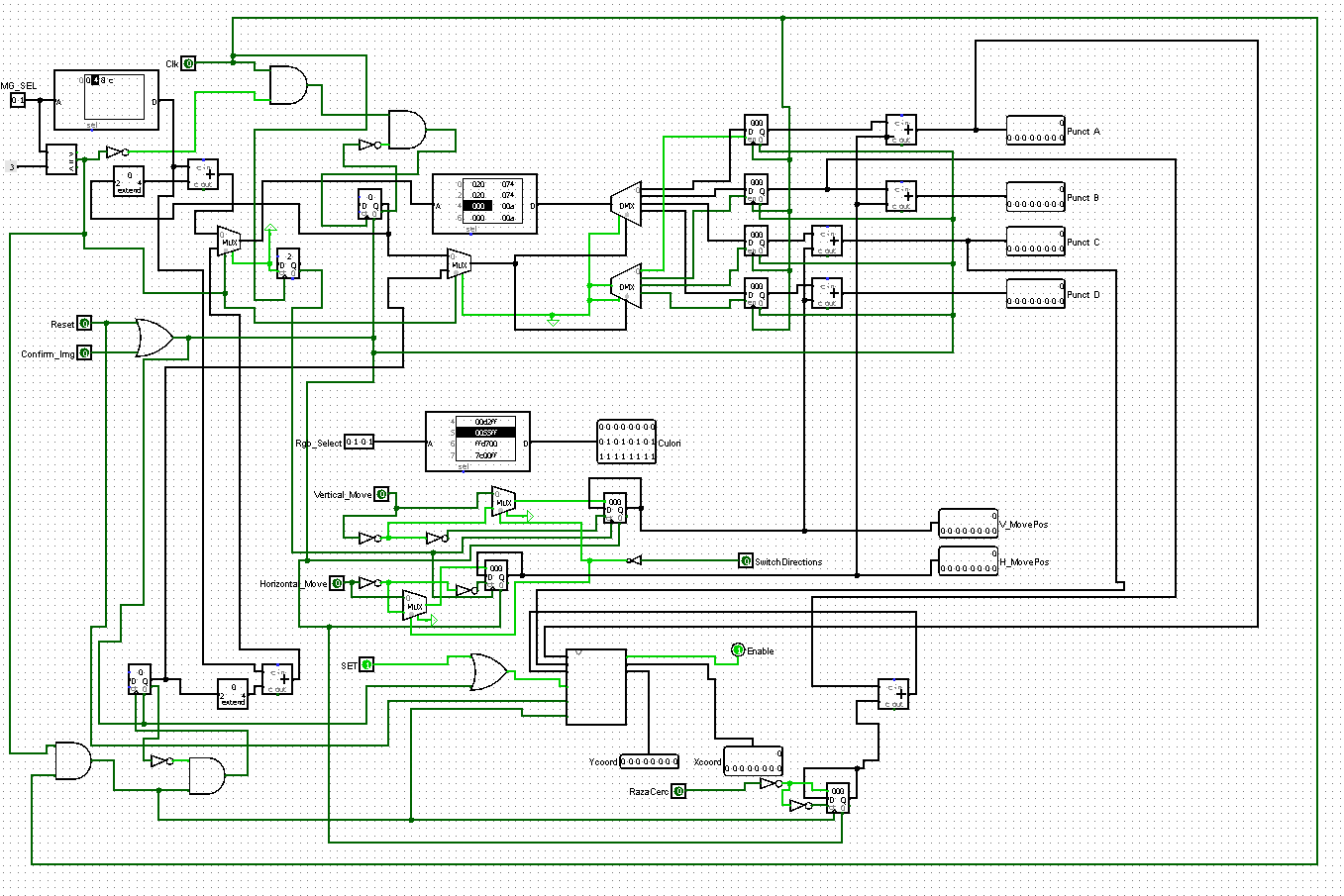


1. Horizontal and Vertical Move: Sunt counterele responsabile pentru miscarea imaginii pe ecran.

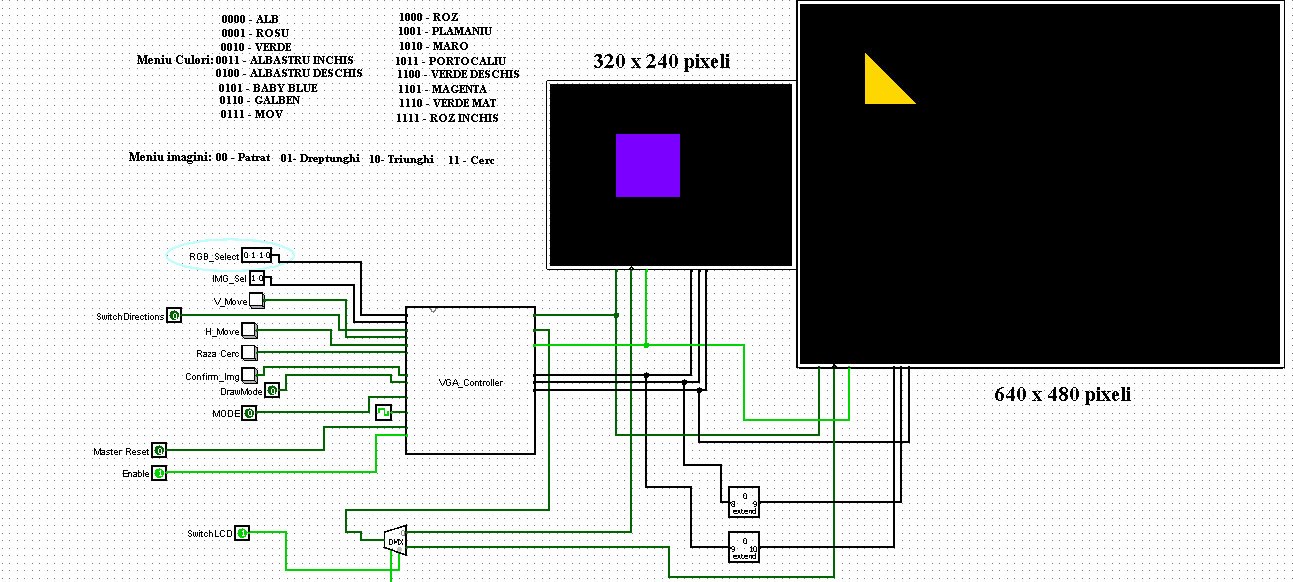
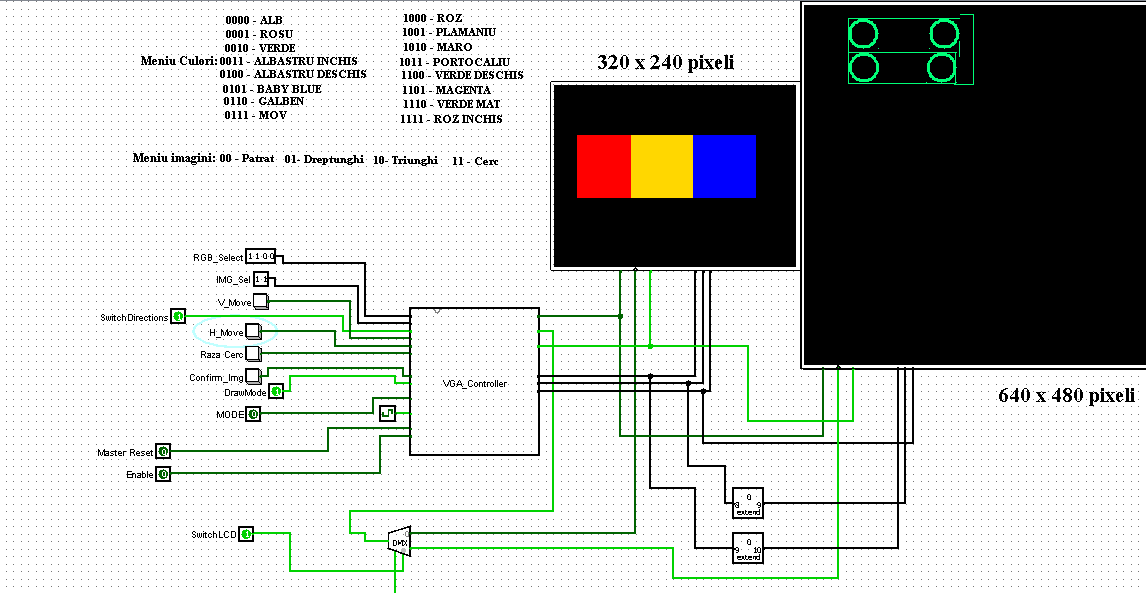


## **2.3.** Prezentare Moduri și Algoritmi de afișare**:**

1. **Mod Afisare:** Acest mod contine mai multi algoritmi precum cel de ***afisare al imaginilor simple*** (pătrat sau dreptunghi), cel de ***afișare al cercului***, cel de ***afișare al triunghiului*** dar și ***algoritmul de mutare al imaginii*** pe ecran.

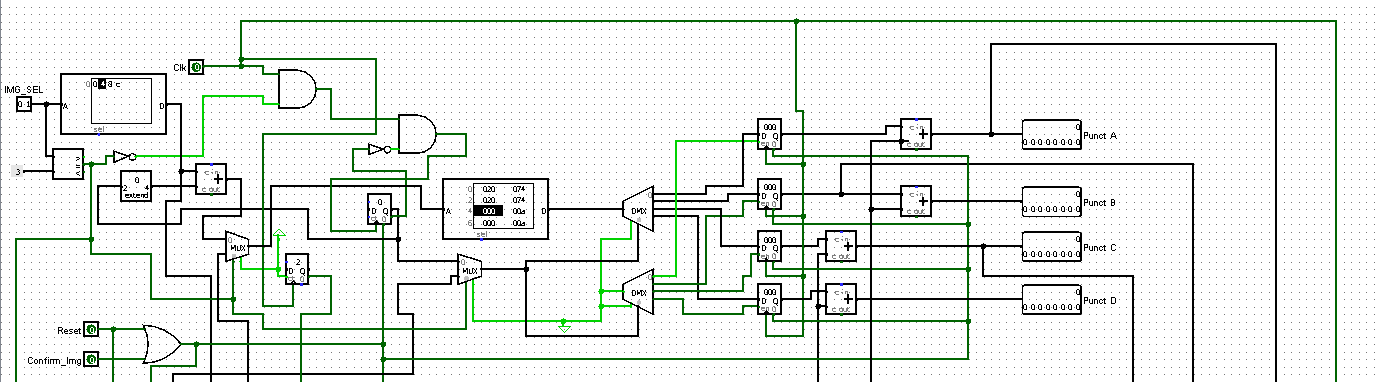
Prezentare generala a schemei pentru **modul Afisare**: 

Prezentare generala a functionalitatii **modului Afisare:**

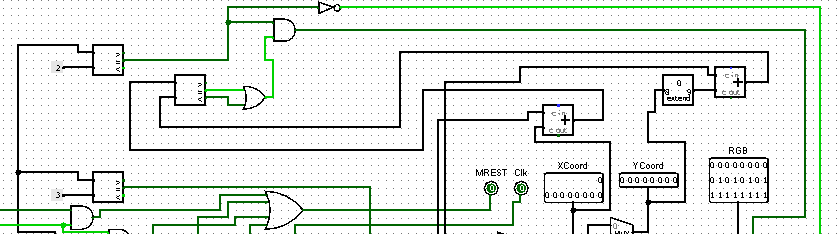


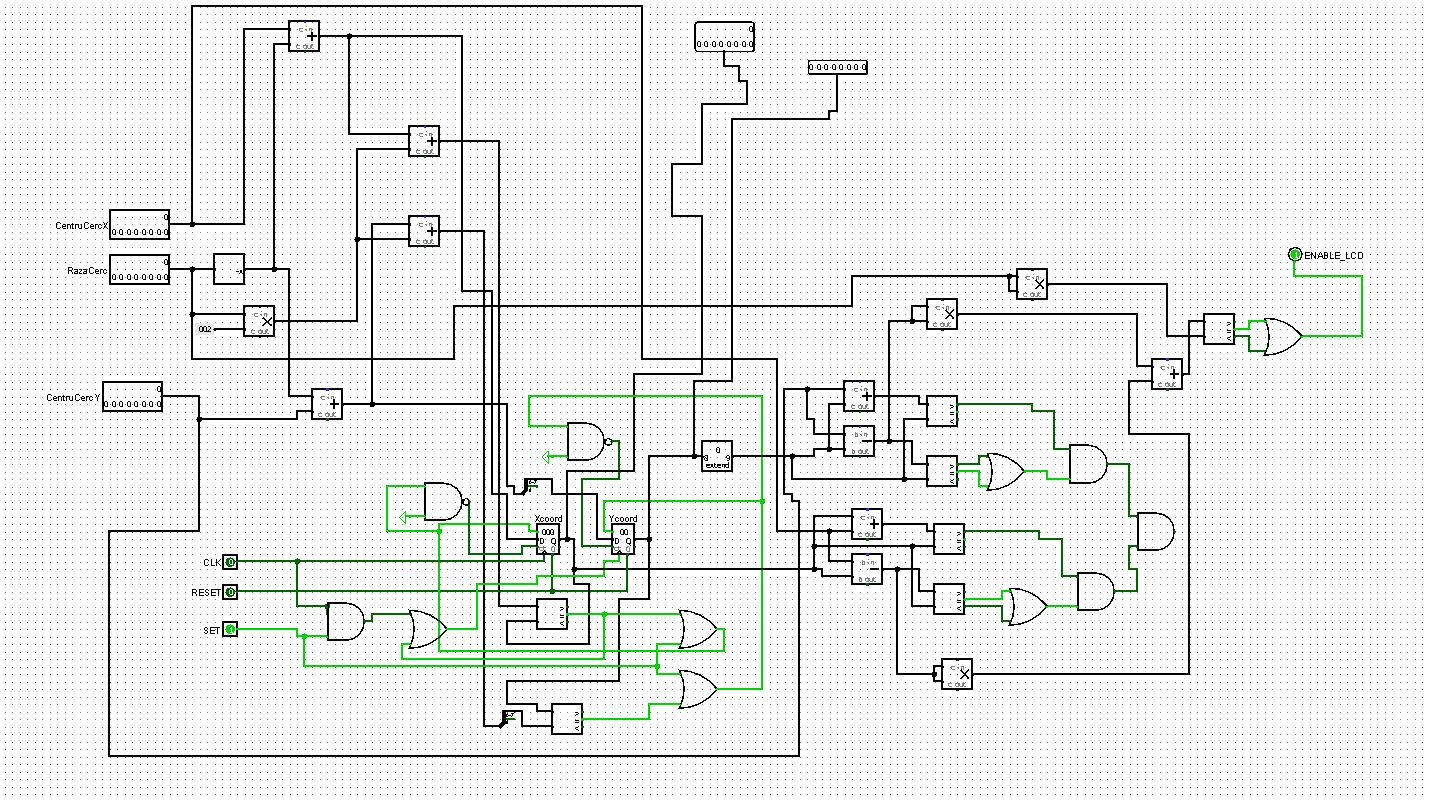
Prezentare algoritmi ce se afla in componenta **modului Afișare**:

* Algoritm afișare imagini din memoria ROM: Acest mod preia coordonatele  varfurilor imaginii din memoria ROM și seteaza culoarea și bucla de numărare pentru countere (H\_counter și V\_counter).

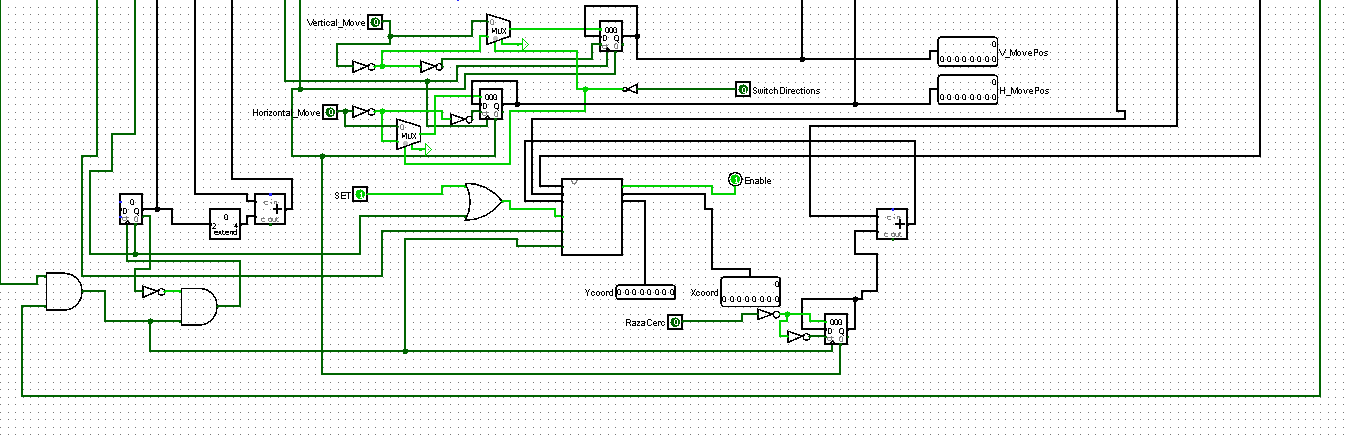


* Algoritm afișare forme precum cerc sau triunghi:
  + Algoritm Triunghi :Folosește formula prezentată la implementarea VHDL care a fost transformată în schema logica.



* Algoritm Triunghi :Folosește formula prezentată la implementarea VHDL care a fost transformată în schema logica.

* Algoritm de mutarea a imaginii pe ecran:

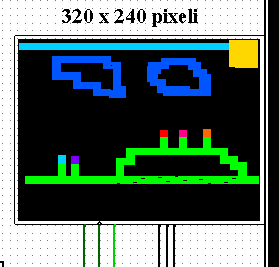


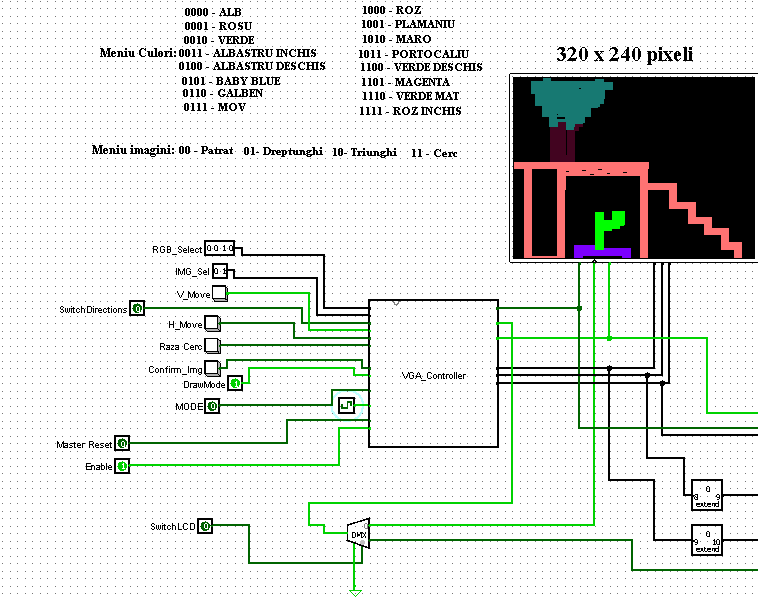
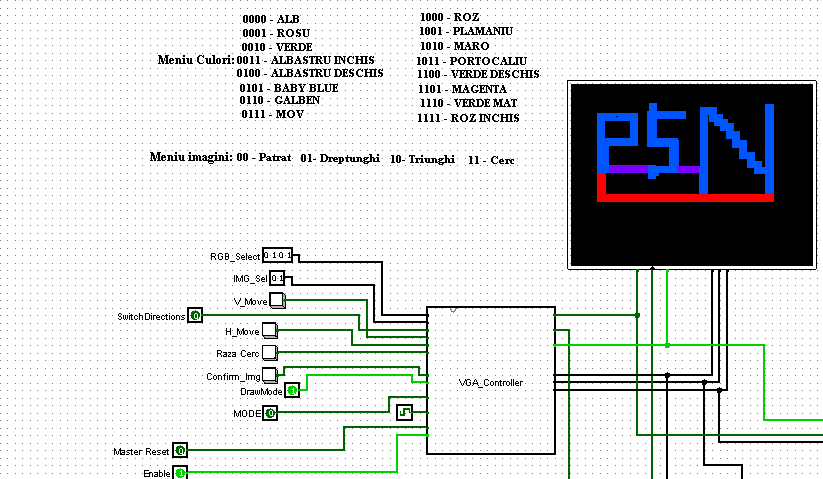
## **2.4.** Modul Paint**.**

Acest mod este unul relativ ușor de implementat și coreleaza cu modul de mișcare a imaginilor pe ecran.

Pentru a putea misca imaginea pe ecran trebuie sa recalculam bucla in care counterele ar trebui sa numere si sa coloreze pixeli iar restul ecranului resetat inapoi la culoarea neagra de Background. Astfel pentru a crea modul paint este indeajuns sa avem un switch care sa controleze cand pixeli colorati din afara buclei trebuie sau nu resetati la culoarea de background.

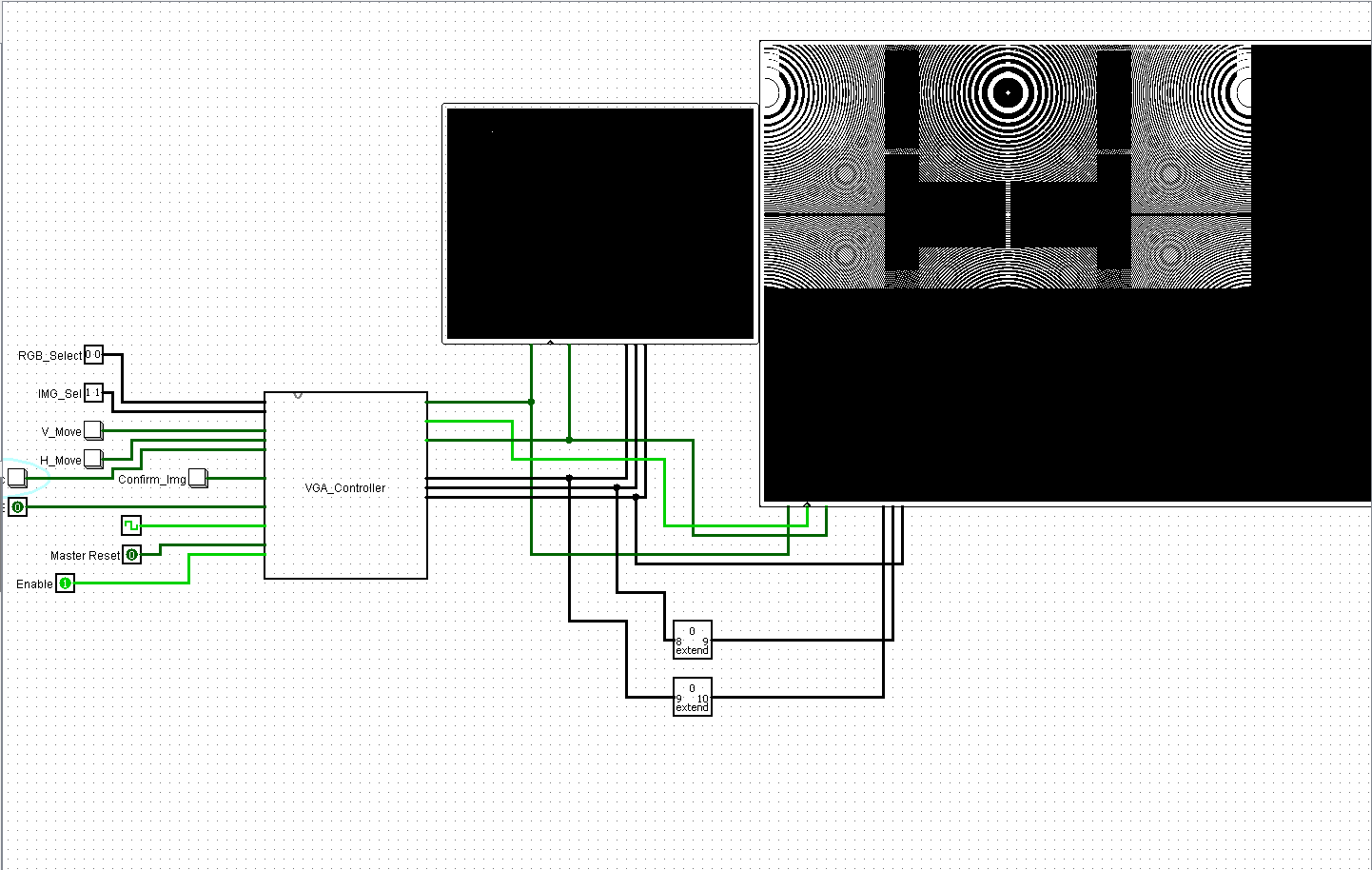
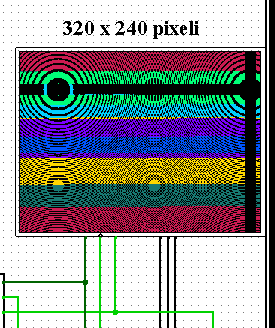
Daca alegem sa nu resetam pixeli din afara buclei nou create vom putea desena pe ecran.

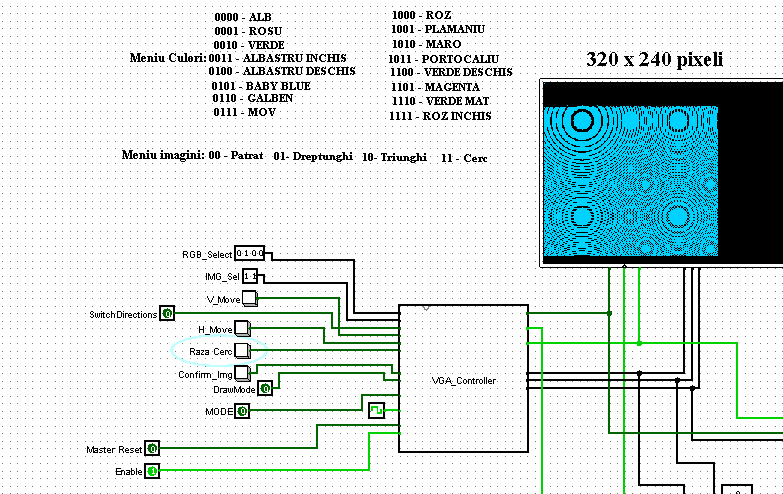
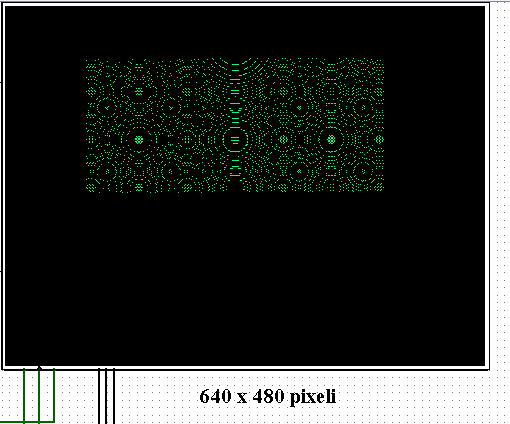


****

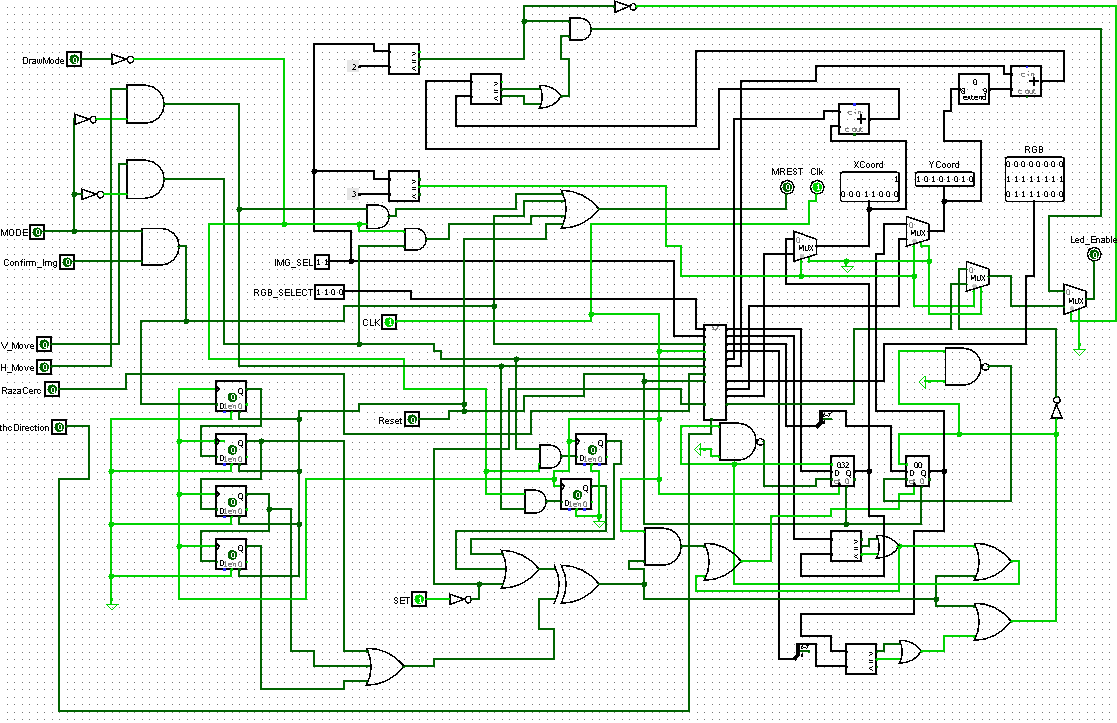
## **2.5.** Modul Abstract**.**

Genereaza forme abstracte prin colorarea cu semicercuri și cercuri în funcție de cum se schimba raza unui cerc pe care il transmitem la VGA Controller.

****



**Prezentare Generala a VGA Controllerului**

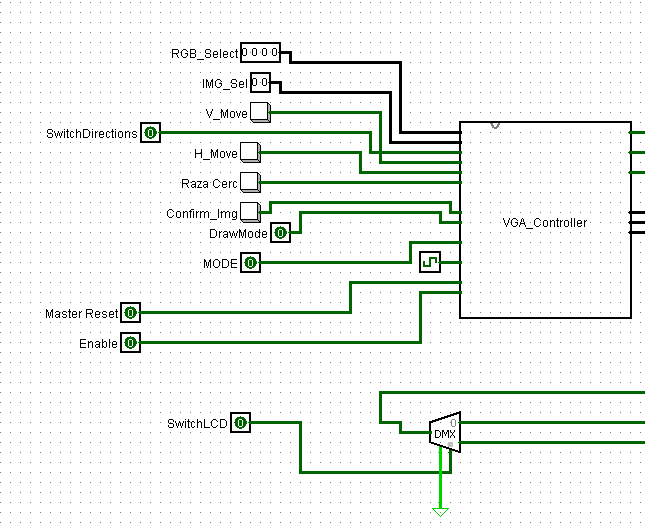
****

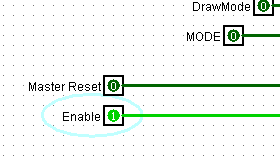
# 3. **Justificarea solutiei alese**

Am ales rezolvarea problemei in acest mod deoarece este unul simplu si optim atat din punct de vedere al memoriei utilizate cat si din punct de vedere al vitezei de generare a imaginilor.De precizat este faptul ca acest mod de implementare desi este foarte avantajos din punct de vedere al vitezei si memoriei are si un dezavantaj, si anume imposibilitatea de generare a unor imagini complexe precum poze.

Am adoptat un stil de implementare comportamental usor de inteles atat de programatori cat si de utilizatori.De asemenea modul de comportare al Vga controller-ului avantajeaza aceasta forma de programare comportamentala.

# **Intructiuni de utilizare si intretinere**

Vga Controllerul contine o serie de optiuni pe care utilizatorul le poate folosi.

In continuare va vom prezenta cum se folosesc acestea:

Pentru ca VGA\_Controllerul sa functioneze trebuie activat prin schimbarea switchului “Enable” din ‘0’ in ‘1’.

Dupa ce acesta a fost activat exista o multitudine de optiuni care pot fi clasate in 3 categorii:

* Selectare/colorare imagine.
* Miscare imagine.
* Free paint.
* **Important!** Selectarea categoriilor se face cu ajutorul switchului **“MODE”**, astfel cand MODE = 1, toate optiunile de la categoria “*Selectare/colorare imagine*” sunt active si pot fi folosite, iar cand MODE = 0 toate optiunile de la categoriile “*Miscare imagine*” respective “*Free paint*” sunt active si pot fi folosite.

## 4.1. Intructiuni de selectare/ colorare imagine (MODE = 1):

* + IMG\_Sel – permite selectarea unei imagini la alegere care urmeaza sa fie tiparita pe ecran doar dupa ce buttonul “***Confirm\_img***” a fost apasat. Acest button este necesar pentru a nu tiparii imaginile in timp ce utilizatorul alege imagine ape care va dori sa o afiseze.

Punem la dispozitia utilizatorului si un meniu care specifica numarul ce trebuie selectat pentru o anumita imagine astfel:



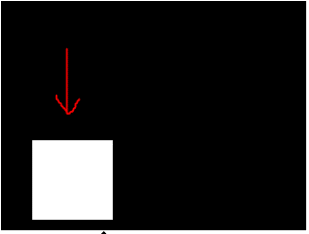
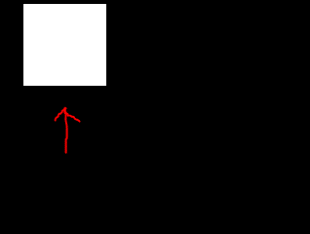
* + RGB\_Select – permite selectarea unei culori in care imaginea urmeaza sa fie colorata. De asemenea oferim un meniu care specifica cum pot fi alese culorile.



**Observatie. Modul** RGB\_Select poate fi folosit si in afara categoriei “*selectare/ colorare imagine”* pentru a usura modul de utilizare.

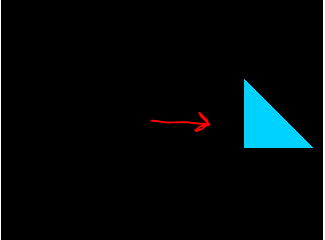
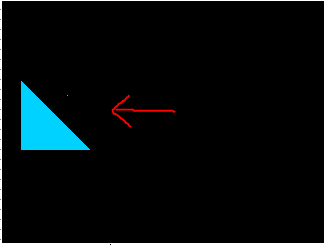
## 4.2. Intructiuni de Miscare imagine(Mode = 0):

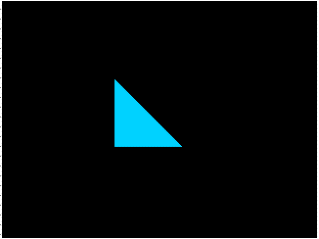
* + V\_Move – reprezinta butonul cu care utilizatorul poate misca imaginea pe axa verticala (sus/jos) in functie de “***SwitchDirections***” astfel :
    - SwitchDirections = 0 => imaginea se deplaseaza in jos.
    - SwitchDirections = 1 => imaginea se deplaseaza in sus.

Exemplu:

**Stare initiala** **SwitchDir = 0** **SwitchDir = 1**

* H\_Move – reprezinta butonul cu care utilizatorul poate misca imaginea pe axa orizontala(dreapta/stanga) in functie de “***SwitchDirections***” astfel:
  + SwitchDirections = 0 => imaginea se deplaseaza in dreapta.
  + SwitchDirections = 1 => imaginea se deplaseaza in stanga.

Exemplu:

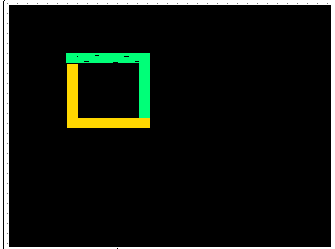


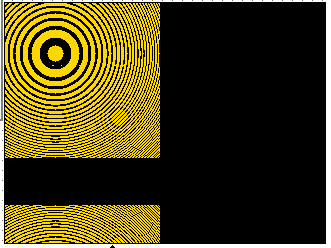
**Stare initiala** **SwitchDir = 0 SwitchDir = 1**

## 4.3. Instructiuni de Free Paint(MODE = 1 and DrawMode = 1):

Pentru a active modul Free Paint este neaparat ca switchul “***DrawMode****”* sa fie active.

* Acest mode foloseste aceleasi instructiuni prezentate mai sus, si anume *“H\_Move*” si “*V\_Move*” precum si “*SwitchDirections*”. In plus fata de modul ***Miscare imagine*** acest mod nu decoloreaza biti in care o anumita imagine se afla la un moment trecut de timp astfel oferind utilizatorului posibilitatea de a crea formele dorite pe ecran.



* 1. Intructiuni de Abstract Mode: Acest mod a fost creat special pentru afisarea de cerc si este capabil sa genereze modele geometrice abstracte. Acest generari se datoreaza schimbarii in lungima a razei cercului care, impreuna cu limitele de incadrare a cercului, altereaza formula matematica folosita.
  + Raza Cerc – este un buton care mareste dimensiunea razei cercului care urmeaza sa fie colorat, generand astfel forma geometrice abstractizate.

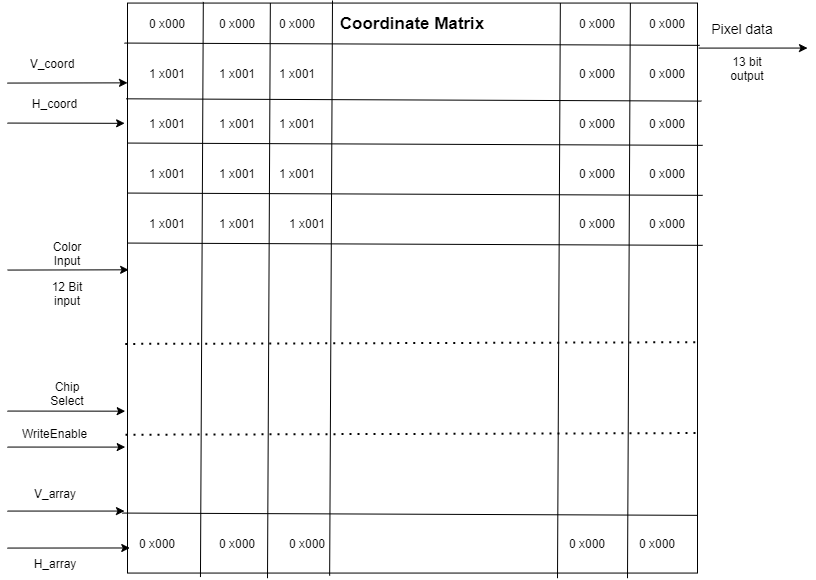
## Alte instructiuni generale:

* + Master Reset – reseteaza ecranul si toate componentele Vga\_Cotroller-ului.
  + SwitchLCD – comuta colorarea imaginii de pe un ecran de 320 x 240 pe un ecran 640 X 480.

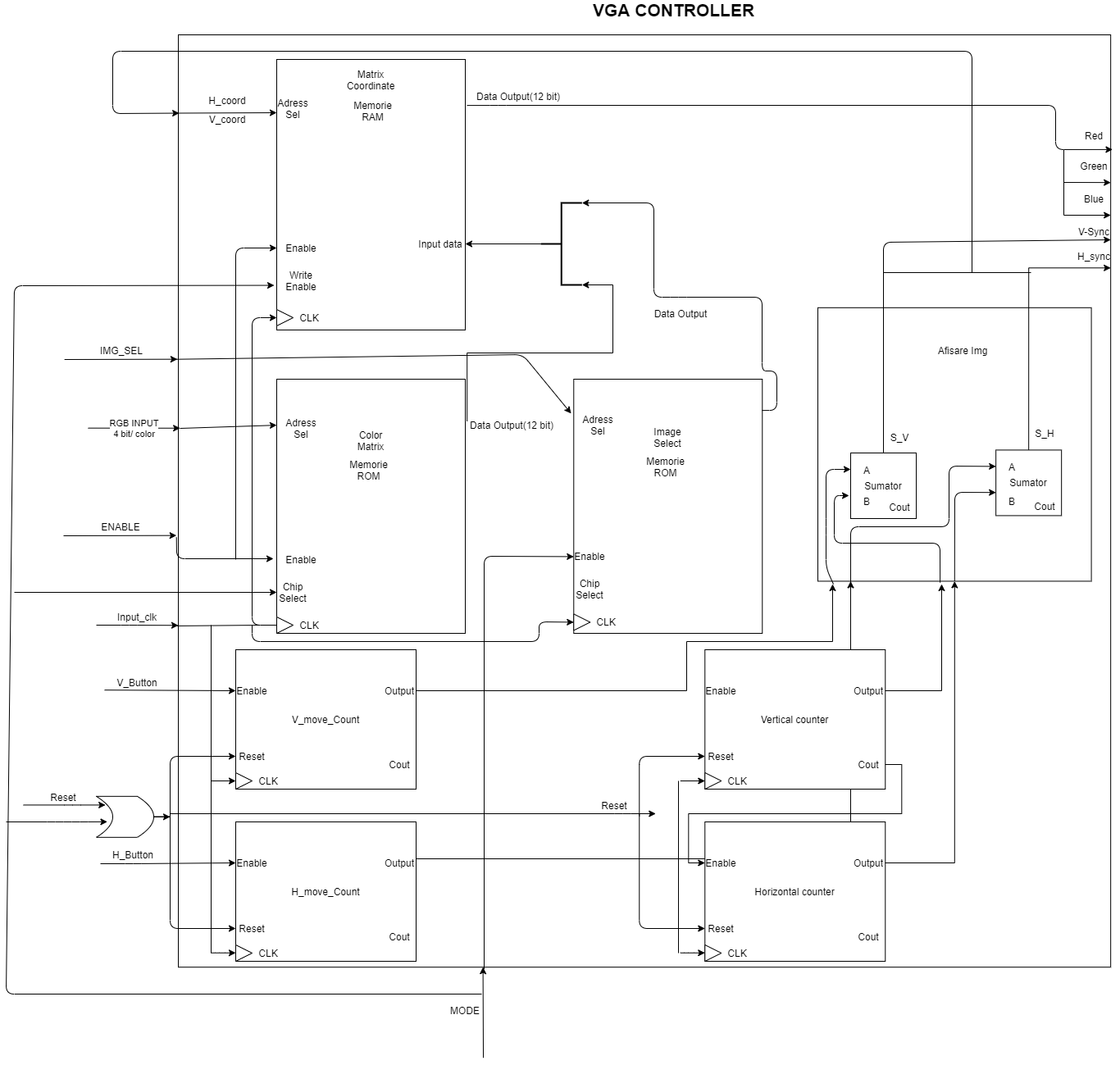
# 5. **Posibilitati de dezvoltare ulterioara**

Pentru un cadru mai mare de dezvoltare al acestui proiect, putem adauga o matrice de coordinate.

Noua componenta are rolul de a genera imagini complexe precum poze prin transmiterea unor seturi de biti si culoarea aferenta acestora.



Astfel, dupa introducerea acestei noi componente arhitectura intregului automat ar fi urmatoarea:



# 6. **Distributie sarcini**:

Parte Proiect “Logisim” – Zaharie Andrei.

Parte Proiect “VHDL” – Jiroveanu Stefan.

Parte Documentatie – Jiroveanu Stefan & Zaharie Andrei.

Parte Vivado – Jiroveanu Stefan & Zaharie Andrei.

# 7. **Bibliografie**:

<https://www.youtube.com/watch?v=4enWoVHCykI>

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

<https://reference.digilentinc.com/learn/programmable-logic/tutorials/vga-display-congroller/start>

<https://www.ece.ucdavis.edu/~bbaas/180/tutorials/vga/>

<https://forum.digikey.com/t/vga-controller-vhdl/12794>

<https://www.youtube.com/watch?v=5exFKr-JJtg>

<https://en.wikipedia.org/wiki/VGA_connector>