

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE



Studenți:

Stolniceanu Iustin-Pavel

Tuchilă Adi-Bogdan

Prof. Îndrumator: Szabolcs - Andras Csillag

Grupa 30217, anul I



Proiect A11: Controller VGA

Documentație

1. Specificația proiectului:

Folosind plăcuțe cu FPGA să se implementeze un controller VGA. Trebuie afișate 4 imagini diferite, selecția fiind citită de la butoanele plăcii. Imaginile trebuie să demonstreze abilitatea de selecție a culorilor (minim 4 culori diferite). De asemenea, poziționarea 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 rezoluții se pot găsi, de exemplu, în tabelul 2-6 din documentația plăcii cu FPGA XUPV2P_User_Guide.pdf, la pag. 37-38. Diagramele de funcționare și explicații se găsesc în manualele de referință ale plăcilor cu FPGA.

Așadar, în cadrul proiectului se va realiza un controller VGA capabil să afișeze 4 imagini diferite, fiecare în 4 culori distincte. De asemenea, pentru fiecare imagine trebuie să existe posibilitatea de o muta pe ecran cu ajutorul butoanelor.

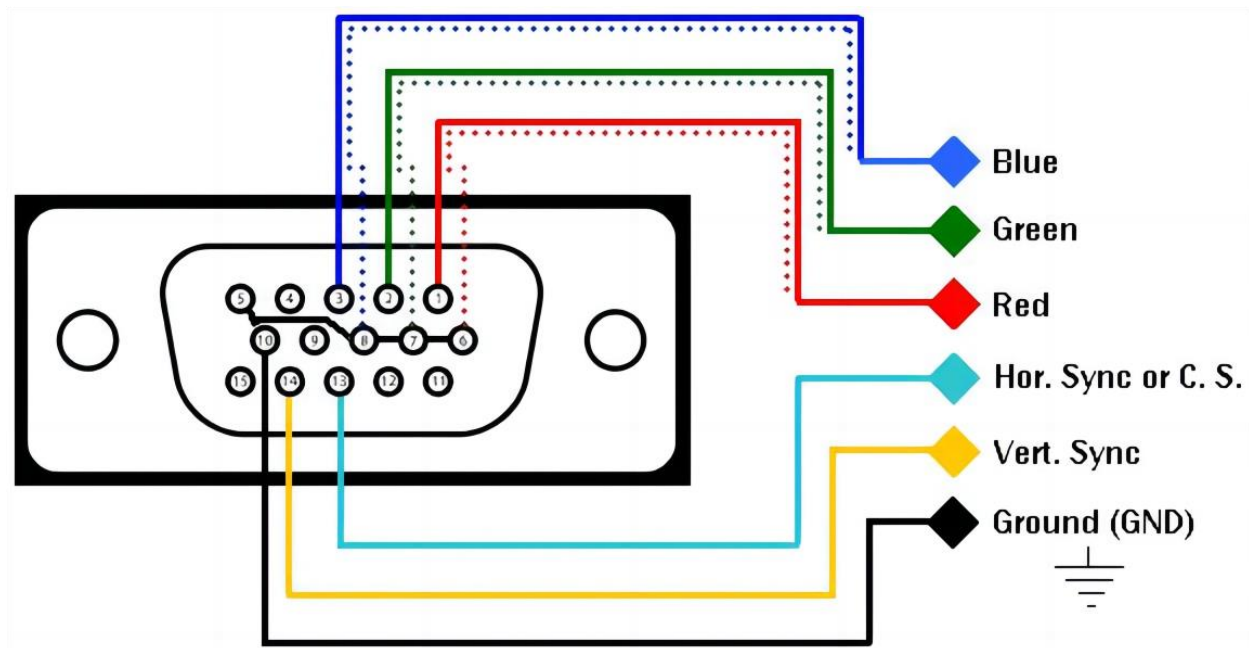
Proiectul este scris în limbajul VHDL, iar sintetizarea și implementarea codului este realizată cu ajutorul programului Vivado.

2. Proiectarea:

În continuare, am început efectiv proiectarea controllerului VGA, în mai multe etape:

2.1. Primii pași

Pentru început, a fost nevoie să ne documentăm cu funcționarea VGA-ului. **Video Graphics Array (VGA)** este un controler de afișare video și un standard grafic, introdus pentru prima dată cu linia de calculatoare IBM PS / 2 în 1987. Conectorul VGA are 15 pini, dintre care trebuie să programăm 5. Deoarece VGA este un standard vechi, acesta este făcut să funcționeze pentru monitoarele cu pistol de electroni, sau CRT. Astfel, programarea unui controler VGA necesită o cunoaștere intimă a tehnologiei care se ascunde în spatele acestor monitoare.



2.1.1. Culoarea Imaginii

Aici folosim 3 pini, Roșu, Verde și Albastru, fiecare culoare având 1 pin. Plăcuța Nexys A7 are nevoie de 4 output-uri pentru fiecare pin, în urma căruia calculează singur rezistențele necesare pentru fiecare pin de culoare. Acestea reprezintă culorile pe care monitorul le afișează. Când trimitem un semnal pe acești pini, monitorul afișează combinația respectivă de culori pe acel pixel. Astfel, dacă vrem să afișăm un pătrat roșu pe monitor, trimitem "1" pe pin-ul "roșu" când

suntem pe suprafața pătratului, "0" în rest. Astfel, afișăm "negru" în jurul pătratului, și "roșu" pe suprafața sa, imaginea rezultantă fiind astfel un pătrat roșu. Dar cum ajungem pe "suprafața" pătratului ?

2.1.2. Poziția pe ecran

Folosim 2 pini, Vertical Sync și Horizontal Sync. Acești pini sunt folosiți pentru a controla poziția pe ecran.

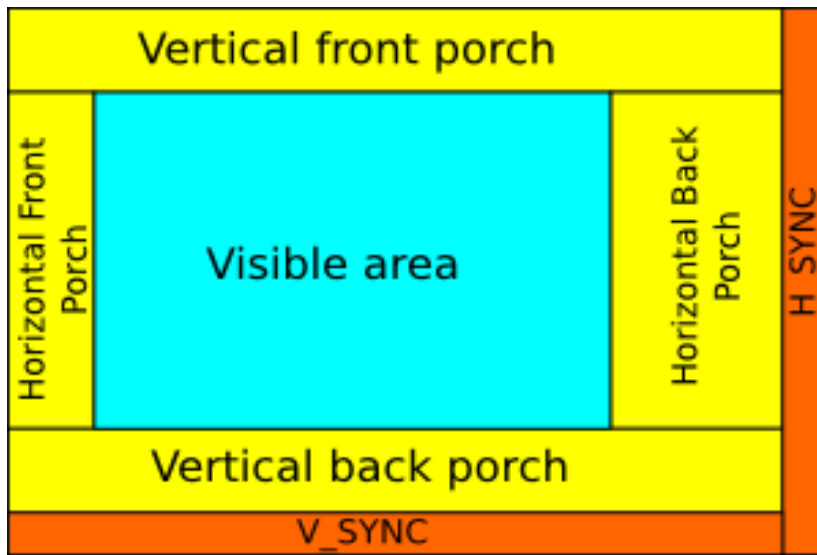
Necesitatea a doua semnale de sync vine de la originile portului VGA, care în trecut era folosit pentru afișarea pe monitoare cu tub. Acest tub mergea orizontal pe ecran, până când primea un semnal de Horizontal Sync, care îi spunea să meargă cu o linie mai jos. Semnalul de Vertical Sync îi spunea tubului să meargă la începutul ecranului. De asemenea, exista și zone invizibile ale ecranului pe care trebuie să le luăm în considerare.

Astfel, trebuie să îi spunem următoarele lucruri monitorului:

- ce culoare să afișeze pe pixelul curent
- când să treacă la următorul rând de pixeli
- când să treacă la un frame nou

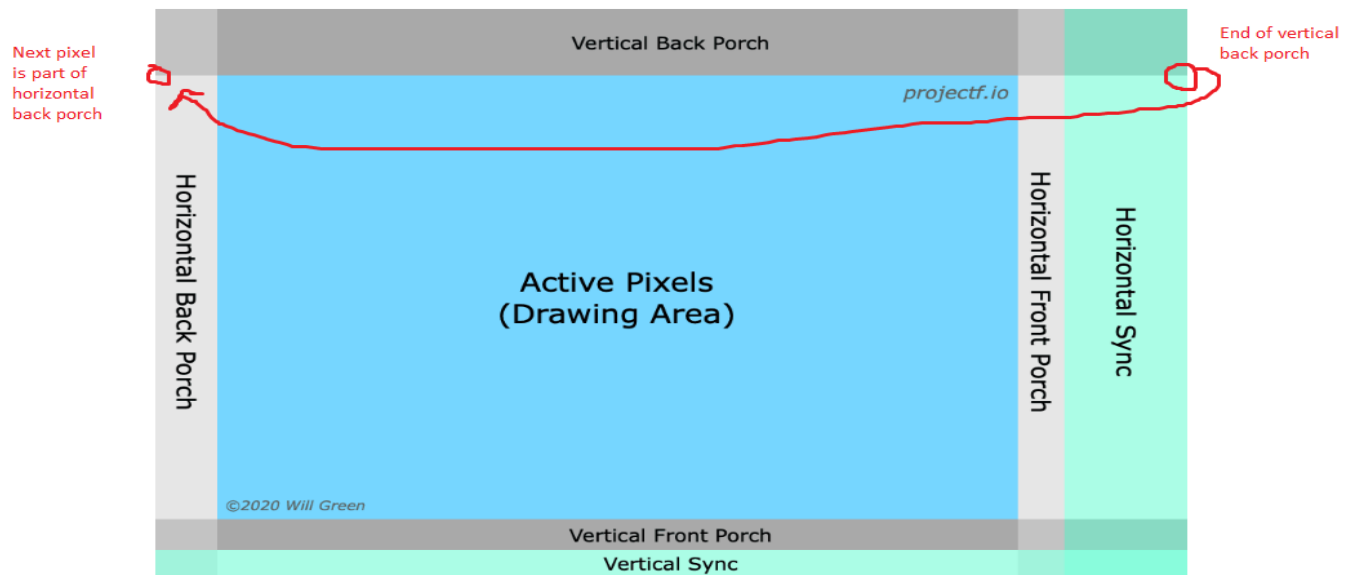
Monitorul trece de la un pixel la altul pe orizontală automat. Noi trebuie să ținem cont intern, pe plăcuța FPGA, la ce pixel suntem, folosind 2 numărătoare. În funcție de pixelii la care suntem, trebuie să trimitem semnale pe HSync și VSync.

În continuare vom numi iteratorul care parcurge pixelii monitorului “pistol de electroni”, întrucât așa se numea la monitoarele cu tub.



Horizontal Sync : “Pistolul de electroni” parcurge întregul monitor (mai exact rezoluția pe care am setat-o. Monitorul știe rezoluția pe care am setat-o după clock-ul pe care îl folosim. Fiecare rezoluție necesită altă frecvență a semnalelor). Acesta "începe" din colțul din stânga sus, și se "mișcă" spre stânga. Când poziția sa ajunge în limita din dreapta, acesta se întoarce în partea stângă, pe rândul următor. În timpul în care acesta se "întoarce" (de ordinul milisecundelor) avem o zonă moartă unde nu se poate afișa nimic. Această zonă poartă numele de Horizontal Back Porch și Horizontal Front Porch. În proiectarea controlerului trebuie să ținem cont de aceste zone moarte.

Vertical Sync trebuie activat când “pistolul de electroni” termină de parcurs întregul monitor (când acesta se află în colțul din dreapta jos al monitorului). Odată activat, pistolul de electroni se întoarce la poziția inițială și începe din nou parcurgerea. De asemenea, timpul pierdut în timpul transmiterii semnalului cauzează zone moarte, pe care trebuie să le luăm în considerare în cod.



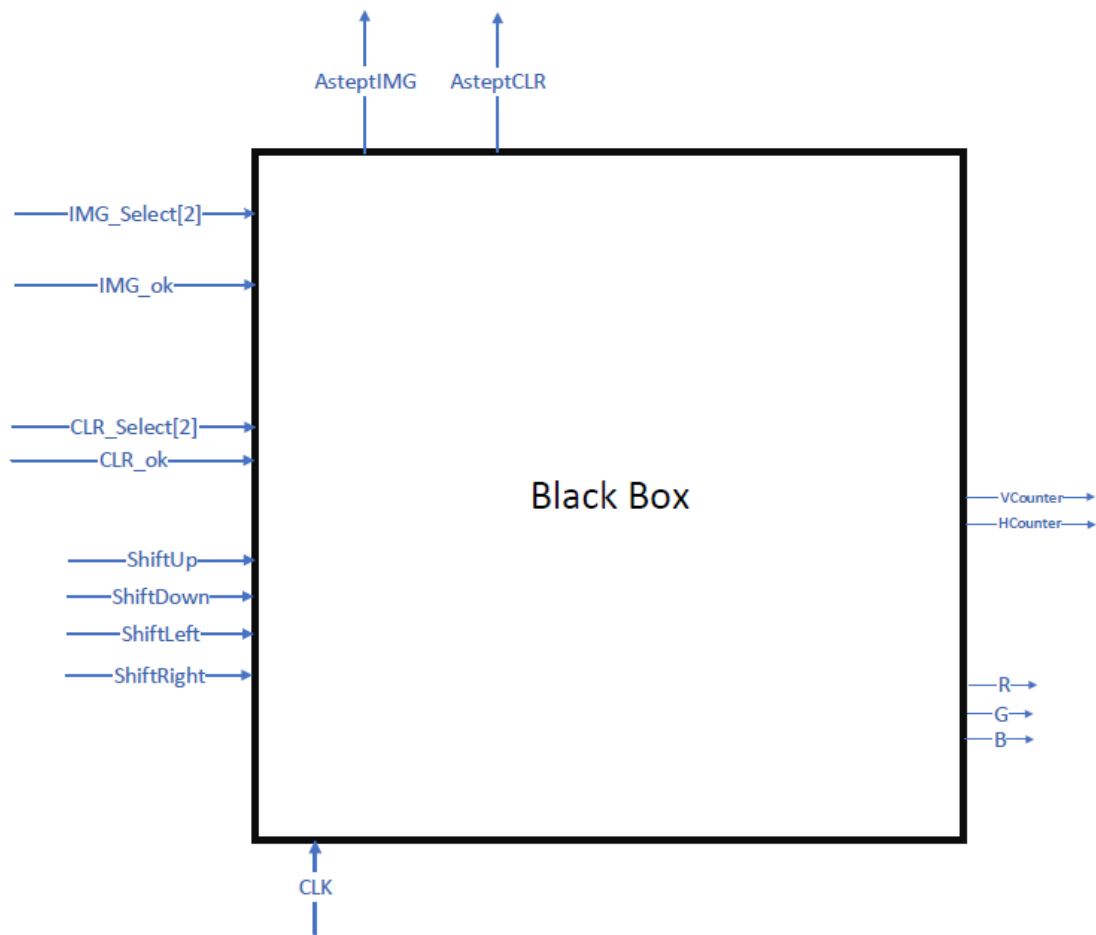
2.2. Design

Pentru început am schițat câteva posibile componente, de care o să avem nevoie :

- 1) Pentru a controla semnalele H-Sync și V-Sync avem nevoie de 2 numărătoare și o componentă care să verifice în ce zonă suntem.
- 2) Pentru a stoca imaginile pe care o să le afișăm folosim 4 memorii VRAM (Video RAM).
- 3) Pentru a selecta imaginea folosim un MUX.
- 4) Pentru a selecta culoarea folosim un MUX care are intrarea SEL Imaginea.
- 5) Stocăm informația pentru culori într-o memorie ROM.
- 6) Intrări pentru a selecta imaginea, culoarea pe switchuri.
- 7) Intrări pentru shift up, down, left, right pe butoane.

2.3. Schema Bloc

Aici găsim **cutia neagră**, adică o privire de ansamblu asupra proiectului. Aici am scos în evidență toate intrările și ieșirile necesare funcționării Controlerului VGA.



În continuare este o listă cu **intrările**, și ce reprezintă fiecare

- 1) IMG_Select[2] = Reprezintă selecția imaginii, pe 2 biți.
- 2) IMG_ok = Un semnal cu care confirmam ca am selectat imaginea.
- 3) CLR_Select[2] = Reprezintă selecția culorii, pe 2 biți.
- 4) CLR_ok = Un semnal care confirma ca am selectat culoarea.

5) Shift Up, Down, Left, Right = 4 semnale, activate de butoane care au rolul de a muta imaginea pe 2 axe.

6) CLK = Semnalul de ceas

Urmează o lista cu **ieșirile**, si o scurta descriere pentru fiecare

1) AsteptIMG = O ieșire care semnalează utilizatorului ca imaginea nu a fost încărcată încă.

2) AsteptCLR = O ieșire care semnalează utilizatorului ca imaginea nu a fost încărcată încă.

3) AfisezIMG = O iesire care semnaleaza faptul ca iesirile pe VGA sunt active

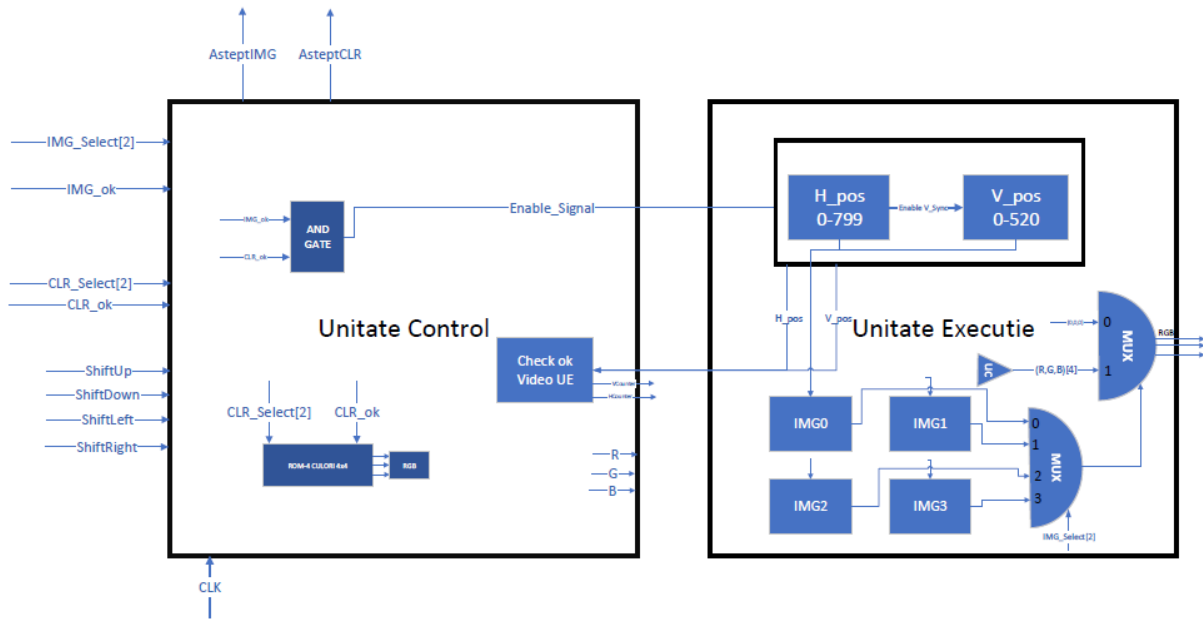
4) VCounter, HCounter = Vsync si HSync

5) RGB = Culorile

2.4. Unitatea de control si Unitatea de execuție

După ce am stabilit toate intrările și ieșirile, următorul pas este împărțirea acestora unității potrivite.

Unitatea de control are “responsabilitatea” de a primi intrările de la utilizator (mai exact imaginea si culoare), dar si de a transmite spre unitatea de execuție semnale specifice.



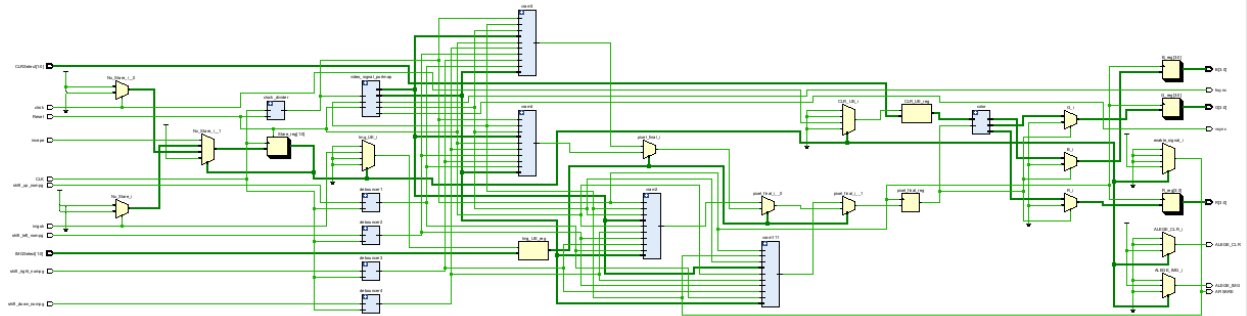
Unitatea de execuție stochează imaginile, iar in aceasta găsim si numărătoarele pentru HSync si VSync, precum si multiplexoarele care selectează imaginea si culoarea.

2.5. Componentele unității de control

Unitatea de control are următoarele componente importante:

- 1) Porți logice care controlează starea circuitului, semnalând unității de execuție când poate sa afișeze imaginea, care imagine si cu ce culoare.
- 2) Debouncere pentru butoane(Semnalele de shiftare), care sunt trimise apoi la UE.

2.6. Componentele unității de execuție



Unitatea de execuție are următoarele componente importante:

- 1) Numărătoare pentru poziția orizontală și verticală, pe care le vom numi de acum hpos și vpos. Când hpos ajunge la maximum sau, acesta va trimite un semnal de enable pentru numărătorul vpos.
- 2) O componentă care verifică în ce zonă a monitorului suntem (moarta/vizibilă/de sincronizare), care are ca intrare poziția orizontală și verticală și ca output HSync, VSync și un semnal de enable_signal, pentru a activa memoria VRAM.

- 3) Un divizor de frecvență, care primește clock de 100 de MHz și scoate un pixelclock, pe care îl stabilim în funcție de rezoluție. În cazul nostru, ne trebuie un clock de 74.250 MHz, obținut de la componenta clock_wizard, din Vivado. Nu putem folosi un divizor de frecvență clasic, cu numărător, deoarece avem nevoie de un clock cât mai “curat”

Pentru simplitatea implementării, am combinat componenta 1 și componenta 2 într-o singură componentă numită video signal:

OK_video este 1 când suntem în zonă vizibilă a monitorului, 0 în caz contrar. Celelalte sunt self explanatory

```

entity video_signal is
  port (Enable : in  STD_LOGIC;
        Rst    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        Vpos   : out STD_LOGIC_VECTOR(11 downto 0);
        Hpos   : out STD_LOGIC_VECTOR(11 downto 0);
        hsync  : out std_logic;
        vsync  : out STD_LOGIC;
        ok_video : out std_logic);
end entity;

```

- 4) Video ROM: Este doar o memorie ROM, de marime 4x12 Biti, care primește input de la unitatea de control si are ca output pinii de culoare ai portului VGA.

```

USE IEEE.STD_LOGIC_1164.ALL;

entity ROM_Color is
  Port (
    Adr_Color    : in STD_LOGIC_VECTOR(1 downto 0);
    Enable_Color : in STD_LOGIC;
    R            : out STD_LOGIC_VECTOR(3 downto 0);
    G            : out STD_LOGIC_VECTOR(3 downto 0);
    B            : out STD_LOGIC_VECTOR(3 downto 0)
  );
end ROM_Color;

architecture Behavioral of ROM_Color is
  signal R0 : STD_LOGIC_VECTOR(3 downto 0) := "1111";
  signal R1 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal R2 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal R3 : STD_LOGIC_VECTOR(3 downto 0) := "1111";
  signal G0 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal G1 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal G2 : STD_LOGIC_VECTOR(3 downto 0) := "1111";
  signal G3 : STD_LOGIC_VECTOR(3 downto 0) := "1111";
  signal B0 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal B1 : STD_LOGIC_VECTOR(3 downto 0) := "1111";
  signal B2 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
  signal B3 : STD_LOGIC_VECTOR(3 downto 0) := "1111";

```

5)

- 6) VRAM: 4 la numar, pentru fiecare imagine

Fiecare VRAM este o memorie RAM de 45x80 biti. Memoria VRAM

precizeaza doar daca pixel-ul curent are sau nu culoare. Primeste 2 adrese: hpos si vpos. Intrucat rezolutia pe care o folosim este 1280x720, trebuie sa impartim numerele primite la 16 cand ne adresam memoriei VRAM.

Mai are 4 inputuri, pentru shiftare

Enable : primeste de la unitatea de control.

Enable_Vram: De la componenta 2.

```
entity VRAM_1 is --primeste pixel clock
  port (hpos      : in  STD_LOGIC_VECTOR(11 downto 0);
        vpos      : in  STD_LOGIC_VECTOR(11 downto 0);
        shift_up   : in  STD_LOGIC;
        shift_down : in  STD_LOGIC;
        shift_right : in  STD_LOGIC;
        shift_left  : in  STD_LOGIC;
        pixel      : out STD_LOGIC;
        clk        : in  std_logic;
        enable     : in  std_logic;
        enable_video : in  std_logic;
        rst        : in  STD_LOGIC
  );
end VRAM_1;
```

Output-ul tuturor VRAM-urilor este dus in unitatea de control, ca intrari la un MUX 4:1, care are ca selectie cei 2 biti selectati de utilizator la inceput(AlegeIMG). Outputul acestui MUX este pus ca selectie intr-un alt MUX, care are ca intrari culorile din ROM-ul de culori si ca output are pinii R(4 biti), G(4 biti), B(4 biti), care sunt trimisi direct la placuta. Astfel, VRAM-ul este de dimensiuni reduse. Daca am fi stocat si informatia de culoare in VRAM, am fi depasit resursele placutei.

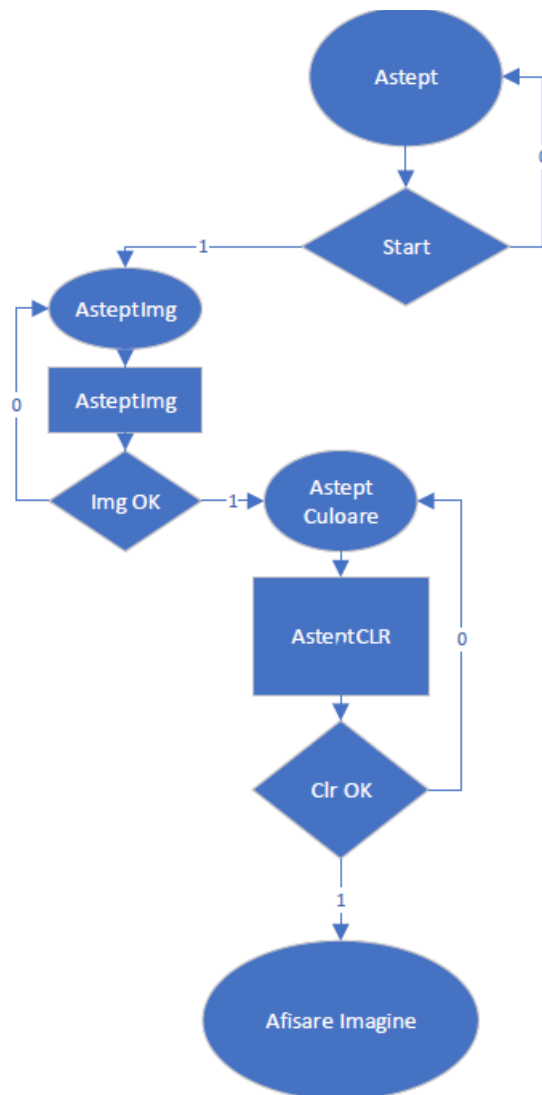
2.7. Organigrama

Organigrama are următoarele 3 stări distincte:

1) *Aștept CLR*

2) *Aștept IMG*

3) *Afișează IMG*



Declanșarea semnalului de *Start* duce în prima stare : **AșteptIMG**. Aceasta stare așteaptă ca utilizatorul să selecteze o imagine folosind intrarea

"SelectImage[2]". In aceasta stare, LED-ul specific pentru ieșirea "AsteptIMG" este aprins. Odată ce imaginea a fost selectata, utilizatorul computează switchul corespunzător pentru intrarea "IMG_ok", fapt care duce circuitul in următoarea stare.

A doua stare, **AsteptCLR**, este similara cu prima, unitatea de control așteaptă ca utilizatorul sa selecteze culoarea dorita, după care aceasta așteaptă ca switchul "CLR_ok" sa fie comutat, fapt care duce la următoarea stare. In aceasta stare LED-ul "AsteptCLR" este aprins.

Ultima stare, AfisareIMG nu necesita intrări obligatorii de la utilizator. In aceasta stare imaginea este afișată pe ecran, iar utilizatorul poate shifta imaginea cu ajutorul butoanelor.

3.0. Justificarea soluției alese

Metoda cu numărătoare a fost cea mai eficienta soluție la care ne-am gândit pentru afișarea imaginilor.

3.1. Probleme întâmpinate pe parcursul proiectului

1) Timp de sintetizare foarte lung : Sintetizarea proiectului durează ~15 minute, fapt care face modificarea codului pentru testare/debugging complicata, înainte de sintetizare codul trebuie verificat riguros.

2) Mărimea mare si limitările NEXSYS A7. Memoriile RAM in care sunt stocate cele 4 imagini, precum si memoria ROM pentru culori ocupa foarte multe resurse ale plăcuței NEXSYS A7, vivado ne-a anunțat de mai multe ori ca depășim resursele plăcuței. Astfel, am redus memoria VRAM la 45x80, pe cand rezolutia pe care o afisam este de 1280x720.

3) Lipsa de monitor VGA. Din păcate singurul monitor VGA de care dispuneam se afla in laboratorul de PSN. Deși aveam plăcută, singura data când puteam testa codul fizic era odată pe săptămâna in timpul laboratorului, fapt care a încetinit procesul de testare si debugging foarte mult.

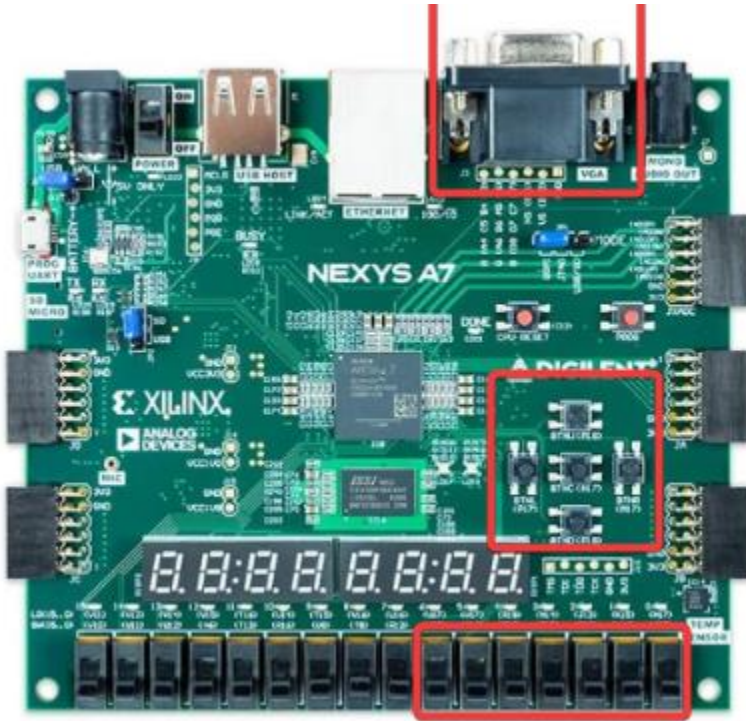
3.2. Ce am învățat din aceste regrese.

1) Deseori verificam codul de 4,5 ori înainte sa sintetizam.

2) Din cauza limitării de resurse, am reușit sa creem o versiune mult mai eficienta a codului decât ce aveam inițial.

3) Am învățat sa facem si sa folosim Testbranch-uri complexe.

4. Manual de utilizare si de întreținere



Se conecteaza monitorul la placa FPGA.

Se folosesc switch-urile, de la dreapta la stanga, astfel:

Se activeaza primul switch. Primul se va aprinde, semnificand faptul ca se asteapta un alt input de la utilizator pentru a selecta imaginea:

- niciun switch activat: un patrat
- al 2-lea switch activat, primul dezactivat: linii verticale
- primul switch activat, al doilea dezactivat: linii orizontale
- ambele switch-uri activate: patrute mai mici care se repeta

Dupa selectia imaginii, se activeaza cel de-al 4 lea switch. Al doilea led se aprinde, semnificand faptul ca se asteapta un input pentru culoarea imaginii pe al 5-lea si al

6-lea switch:

-niciun switch activat: rosu

-al 2-lea switch activat, primul dezactivat: verde

-primul switch activat, al doilea dezactivat: albastru

-ambele switch-uri activate: alb

Dupa selectia culorii, se activeaza cel de-al 7 lea switch. Un al 3-lea led se va aprinde si imaginea va fi afisata pe monitor. Pentru mutarea imaginii pe monitor, se folosesc butoanele de pe placa FPGA.

5.Posibilitati de utilizare ulterioara

Adăugarea de imagini si culori.

6. Bibliografie

<https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual>

https://acg.cis.upenn.edu/milom/cis371-Spring13/lab/XUPV2P_User_Guide.pdf

https://en.wikipedia.org/wiki/Cathode-ray_tube

https://en.wikipedia.org/wiki/Video_Graphics_Array

<http://tinyvga.com/>