DEPARTAMENTUL DE CALCULATOARE

UTILIZAREA PORTULUI VGA AL PLĂCII BASYS 3 PENTRU AFIȘAREA UNOR IMAGINI

David Harabagiu & Diana Bejan
30239 & 302310
Prof. Coordonator Dan Butiri

May 8, 2018

Conținut

1 Rezumat

Acest document prezină o scurtă sinteză cu privire la realizarea unui circuit numeric pe o placă de dezvoltare Basys 3 FPGA, capabil să utilizeze portul VGA al acesteia pentru afișarea unor imagini pe un display. Problema principală este cea a conducerii semnalelor portului VGA, astfel încât acestea să fie sincronizate corect. Alte obiective cuprind manipularea culorilor cu ajutorul unor filtre de imagini ce pot fi schimbate de către utilizator cu ajutorul unor comutatoare și crearea unei aplicații interactive care să determine ieșirea în funcție de stimulii utilizatorului. Pentru dezvoltare, a fost ales limbajul VHDL și mediul de dezvoltare Xilinx Vivado 2017.4. Un alt utilitar folosit, de origine proprie, este o aplicație ce generează, pe baza unei imagini un modul VHDL ce reprezintă un look-up table cu informațiile de culoare ale imaginii. Pentru versionarea codului sursă a fost utilizat Git. Rezultatul este o aplicație versatilă ce are un potențial ridicat de dezvoltare ulterioară în lumea noastră de consumatori media.

2 Introducere

Video Graphics Array (VGA) este hardware-ul de afișare ce a fost introdus împreună cu seria de calculatoare IBM PS/2 în anul 1987, succesor al CGA și EGA introduse în calculatoarele personale IBM dezvoltate anterior. VGA a fost ultimul standard grafic IBM căruia i s-au conformat majoritatea producătorilor de calculatoare, făcându-l standardul fundamental ce tot hardware-ul grafic post-1990 ar fi trebuit să îl implementeze. Astăzi, interfața VGA analog este utilizată pentru video la calitate înaltă, inclusiv rezoluții 1080p sau mai mari. [6]

VGA este considerat mai mult un "Array" decât un "adaptor" deoarece a fost implementat de la început ca un singur cip ce a înlocuit atât generatorul de adrese video Motorola 6845 cât și alte zeci de circuite logice discrete ce acopereau complet plăcile ISA ale MDA, CGA și EGA. Implementarea pe un singur cip a permis ca VGA să fie amplasat direct pe placa de bază a unui PC cu dificultate minimă, deoarece erau necesare doar o memorie video, oscilatoare și un RAMDAC extern. Așadar, primele modele IBM PS/2 erau echipate cu VGA pe placa de bază, în contrast cu modele mai vechi ce necesitau un adapotor de afișare instalat într-un slot pentru a conecta un monitor. VGA folosește conectorul DE-HD15. [6]

Obiectivul acestui proiect este de a demonstra funcționalitățile VGA, prin implementarea unui controller pe o placă FPGA (Field-programmable gate array) și utilizarea port-ului plăcii pentru a afișa imagini pe un monitor. S-au implementat aplicații cum ar fi afișarea fotografiilor, aplicarea filtrelor de imagini (negativ, controlul contrastului, grayscale, dezactivarea individuală a canalelor Red, Green, Blue), precum și o aplicație interactivă - o implementare a jocului Snake.

În soluția propusă am separat partea de controller VGA de aplicațiile asociate menționate anterior. Controller-ul folosește 2 numărătoare pentru a genera semnalele de sincronizare orizontală și verticală, precum și semnalele de adresare a memoriei video. Implementarea aleasă utilizează modul video VGA 640x480@60Hz. Datorită anumitor limitări hardware (memoria disponibilă pe placa de dezvoltare FPGA), fotografiile stocate au o dimensiune de 160x120, iar apoi sunt redimensionate de către hardware, semnalele digitale de adresare fiind împărțite cu 4 înaintea selectării pixelului din fotografie. Pentru a controla aplicația interactivă, este utilizată comunicarea cu o tastatură PS/2 prin portul disponibil pe placa de dezvoltare.

În secțiunea "Fundamentare teoretică" se prezintă fundamentele teoretice legate de VGA, portul PS/2 și filtrele de imagini. Secțiunea "Proiectare și implementare" prezintă detalii tehnice legate de implementarea aleasă, cum s-a ajuns la această soluție, precum și un ghid pentru configurarea și utilizarea aplicației. În secțiunea "Rezultate experimentale" sunt prezentate informații relevante și rezultate ale simulării și implementării.

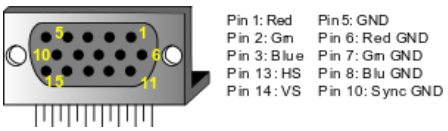

Fig. 1: Conectorul VGA

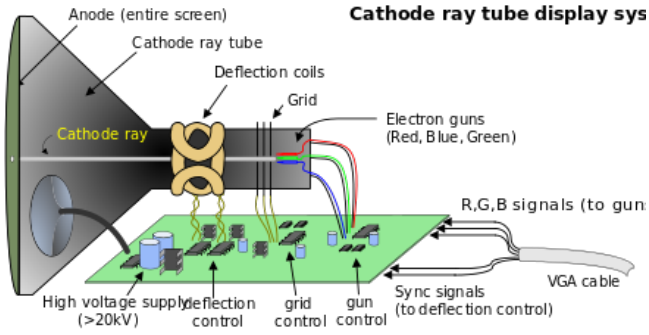

Fig. 2: Monitor CRT Multicolor

3 Fundamentare teoretică

3.1 VGA (Video Graphics Array)

VGA (Video Graphics Array) se referă de obicei la standardele de afișare analog a calculatoarelor, la conectorul DE-15 (cunoscut ca și conectorul VGA) sau la rezoluția 640x480.

Conectorul DE-15 este un conector cu 15 pini amplasați in 3 rânduri. Numele fiecărui pin este specificat în Fig.1. Semnalele Red, Grn, Blue sunt 3 semnale analog ce specifică culoarea unui punct de pe ecran, pe când HS și VS ne dau o referință poziționlă despre unde ar trebui ca punctul să fie afișat. Display-urile VGA bazate pe CRT folosesc raze de electroni modulate de amplitudine ce se mișcă pentru a afișa informații pe un ecran cu peliculă de fosfor. Display-urile LCD folosesc o matrice de comutatoare ce impun un voltaj pe o suprafață mică de cristal lichid. Chiar dacă urmatoarea descriere este limitată la monitoarele CRT, cele LCD au evoluat pentru a utiliza aceiași timpi de semnal ca și cele CRT. Monitoarele CRT multicolor folosesc 3 raze de electroni (pentru roșu, verde și albastru) pentru a energiza pelicula de fosfor de pe partea interioară a ecranului (Fig.2).

Informația este afișată doar atunci când raza se mișcă "înainte" (de la stânga la dreapta și de sus în jos), și nu când raza este resetată în colțul stânga sus a ecranului. Astfel, mult timp potențial pentru afișare este pierdut în perioadele de "blanking", când raza este resetată și stabilizată și începe o nouă trecere orizontală sau verticală. Dimensiunea razei, frecvența cu care raza este trasată de-a lungul ecranului și frecvența la care raza poate fi modulată determină rezoluția.

Display-urile moderne VGA pot acomoda diferite rezoluții, iar un circuit de control VGA dictează rezoluția prin producerea diferitor semnale de timp. Controller-

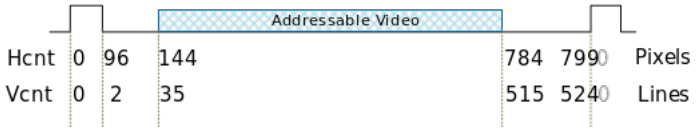

Fig. 3: Generarea HS și VS pe baza valorilor numărătoarelor

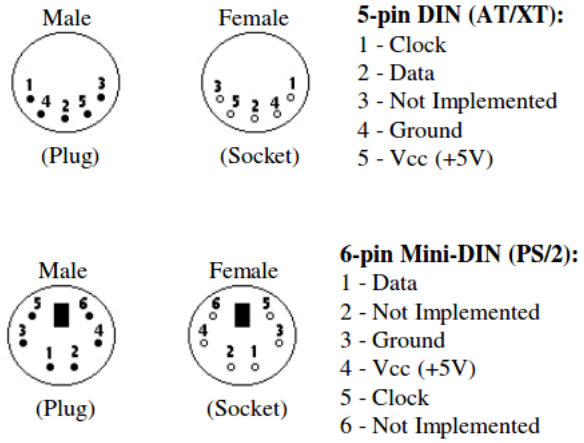

Fig. 4: Configurația pinilor PS/2

ul trebuie să producă pulsuri de sincronizare la 3.3V (sau 5V) pentru a seta frecvența cu care curentul se deplasează prin bobinele de deflecție, și trebuie să se asigure că datele video sunt aplicate la timpul potrivit. Display-urile definesc un număr de "rânduri" ce corespund unui număr de treceri orizontale și un număr de "coloane" care corespund unei suprafețe pe fiecare rând ce este atribuită unui pixel.

Datele video vin de obicei vin de obicei dintr-o memorie video refresh, cu unul sau mai mulți bytes atribuiți fiecărui pixel. Basys 3 utilizează 12 biți pe pixel. Controller-ul trebuie să indexeze într-o memorie video în timp ce raza se mișcă pe ecran, și să aplice informația despre pixel la timpul precis în care raza se mișcă deasupra acelui pixel.

Un controller VGA trebuie să genereze semnalele de timp HS și VS pentru a coordona transmisia de date video pe baza unui semnal de tact "Pixel Clock". Acest semnal de tact definește timpul disponibil pentru a afișa un pixel. Semnalul VS definește rata de "refresh" a ecranului. Numărul de linii de afișat definește rata de "retrace" orizontal. [2]

3.2 Tastatura PS/2

Portul fizic PS/2 poate fi de 2 feluri: cu 5 pini sau cu 6 pini, singura diferență fiind aranjarea pinilor. Configurația pinilor este afișată in Fig.4. Vcc/Ground asigură putere dispozitivului. Liniile Data si Clock sunt amândouă open collector cu rezistențe pullup la Vcc. O interfață open-collector are 2 stări posibile: low și

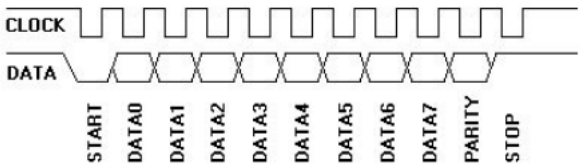

Fig. 5: Comunicarea PS/2 dispozitiv-gazdă

impedanță înaltă. În starea low, un tranzistor trage linia la ground. În starea de impedanța înaltă, interfața se comportă ca un circuit deschis si nu trage linia nici jos nici sus. O rezistență pullup este conectată între magistrală si Vcc astfel încât magistrala este trasă sus dacă niciun dispozitiv de pe magistrală nu o trage jos.

Mouse-ul si tastatura PS/2 implementează un protocol serial bidirecțional sincron. Magistrala este "idle" când amândouă linii sunt sus. Aceasta este singura stare în care dispozitivului îi este permis să transmită date. Gazda este singura ce are control deplin asupra magistralei si poate inhiba comunicarea în orice moment prin tragerea linei Clock jos. Dispozitivul generează mereu un semnal de tact.

Toate datele sunt transmise cu câte un byte pe rand. Fiecare byte este transmis într-un cadru conținând 11-12 biți. Acești biți sunt:

- 1 bit de start, întotdeauna 0;
- biți de date, cel mai puțin semnificativ bit primul;
- 1 bit de paritate (1 dacă numărul de biți de 1 sunt impari);
- 1 bit de stop, întotdeauna 1;
- 1 bit de confirmare (doar la comunicare gazdă-dispozitiv).

Datele transmise de la dispozitiv la gazdă sunt citite pe frontul descendent al semnalului de tact. Frecvența acestui semnal trebuie sa fie între 10 și 16.7 KHz. Când dispozitivul vrea sa transmită date, mai întâi verifică dacă linia Clock e la nivelul logic high. Dacă nu, înseamnă că gazda inhibă comunicarea și dispozitivul va trebui să rețină orice date de trimis până când gazda eliberează linia Clock. Dispozitivul trimite un bit pe linia Data când Clock e high, iar gazda citește când Clock e low.

Procesorul tastaturii monitorizează permanent matricea de taste. Dacă găsește o tastă ce este apăsată, ridicată, sau ținută jos, tastatura va trimite un pachet de informație numită "scan code" către calculator. Există două tipuri de scan code-uri: "make codes" și "break codes". Un make code este trimis atunci când o tastă este apăsată sau ținută jos. Un break code este trimis atunci când tasta este ridicată. Fiecare tastă are propriul make code și break code. Mulțimea de make code-uri si break code-uri a fiecărei taste definesc un "scan code set". [1]

3.3 Filtre de imagini

3.3.1 Filtre de contrast

Primul pas în a calcula factorul de corecție a contrastului este dat de formula:

$$F = \frac{259 * (C + 255)}{255 * (259 - C)}$$

Valoarea C din formula denotă nivelul de contrast dorit. Următorul pas îl constă în ajustarea contrastului, aplicând următoarea formulă fiecărei componente de culoare (R, G, B) din fiecare pixel:

$$newRed = Truncate(F * (Red(color) - 128) + 128)$$

$$newBlue = Truncate(F * (Blue(color) - 128) + 128)$$

$$newGreen = Truncate(F * (Green(color) - 128) + 128)$$

De asemenea, trebuie să trunchiem rezultatul astfel încât acesta să fie între valorile acceptabile (0 și 255). Valoarea C poate fi între -255 și 255. O valoare negativă înseamnă scăderea nivelului de contrast, iar una pozitivă creșterea acestuia. [3]

3.3.2 Filtru grayscale

Pentru a converti o imagine RGB în grayscale, valoarea fiecărui canal de culoare trebuie înlocuită cu media aritmetică a canalelor. [4]

$$C = \frac{R + G + B}{3}$$

3.3.3 Filtru negativ

Negativul unei imagini se poate obține calculând complementul fiecărui canal. [4]

$$R' = 255 - R$$

$$G' = 255 - G$$

$$B' = 255 - B$$

4 Proiectare și implementare

În Fig.6 este prezentată schema bloc a întregului sistem, împreună cu porturile de I/O. Porturile de intrare sunt următoarele:

- CLK100MHZ - Semnalul de ceas al plăcii de dezvoltare Basys 3. Acesta are o frecvență de 100 MHz;
- PS2Clk - Semnalul de ceas al tastaturii, corespunzător pinului Clock al conectorului PS/2 (Fig.4);
- PS2Data - Semnalul de date serială a tastaturii, corespunzător pinului Data al conectorului PS/2 (Fig.4);

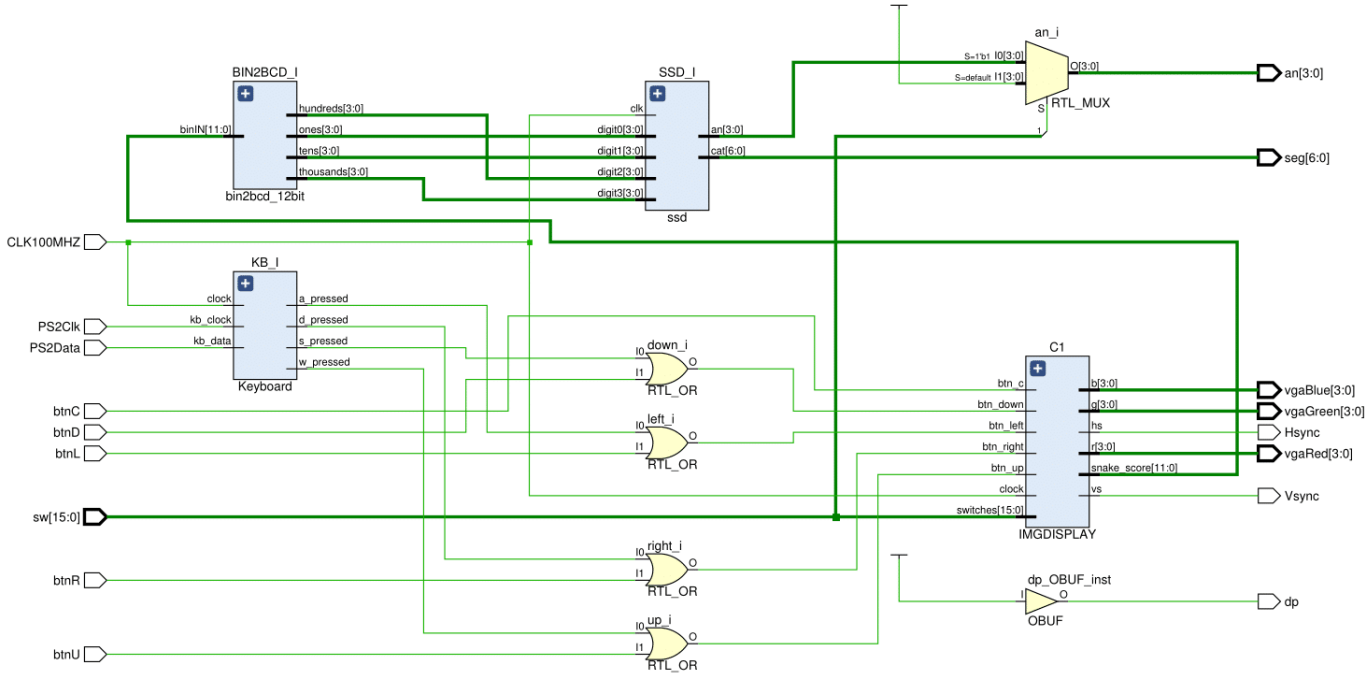

Fig. 6: Arhitectura generală a sistemului

- sw - Comutatoarele plăcii de dezvoltare. Acestea sunt utilizate pentru selecta imaginea afișată, pentru a activa modulul Snake, precum și pentru a alege filtrele de imagini ce vor fi aplicate înainte de afișare;
- btnC - Butonul central al plăcii de dezvoltare. Este utilizat pentru a reseta modulul Snake;
- btnU, btnD, btnL, btnR - Butoanele de sus, jos, stânga, respectiv dreapta ale plăcii de dezvoltare. Sunt utilizate pentru a controla directia șarpelui în modulul Snake.

Porturile de ieșire sunt următoarele:

- HSync - Semnalul de sincronizare orizontală al conectorului VGA. (Fig.1);
- VSync - Semnalul de sincronizare verticală al conectorului VGA. (Fig.1);
- vgaRed - Componenta roșie a pixelului trimis la monitor prin conectorul VGA;
- vgaGreen - Componenta verde a pixelului trimis la monitor prin conectorul VGA;
- vgaBlue - Componenta albastră a pixelului trimis la monitor prin conectorul VGA;
- an - Anozii afișoarelor cu 7 segmente ale plăcii de dezvoltare;
- seg - Catozii afișoarelor cu 7 segmente ale plăcii de dezvoltare;
- dp - Catodul corespunzător punctului afișoarelor cu 7 segmente ale plăcii de dezvoltare. Acesta este mereu '1' logic (punctul este stins).

Afișoarele cu 7 segmente sunt folosite pentru a afișa scorul jucătorului în Snake. Afișoarele sunt dezactivate atunci când modulul Snake este inactiv.

Componenta "IMGDISPLAY" controlează afișarea imaginilor pe display. Aceasta conține controller-ul interfeței VGA, memoria cu imaginile ce pot fi afișate, precum și modulul Snake.

Componenta "Keyboard" citește serial datele de la conectorul PS/2 și detectează apăsarea tastelor W, A, S, D - utilizate pentru a controla șarpele.

Componenta "bin2bcd_12bit" realizează conversia unui număr unsigned pe 12 biți la un număr în reprezentarea BCD.

Componenta "ssd" se ocupă de afișarea numarului BCD pe display-ul cu 7 segmente.

4.1 Afișarea imaginilor

Pentru afișarea imaginilor, modulul "IMGDISPLAY" conține componenta "Vga_Controller", care primește semnalul PixelClock (semnal de tact cu frecvența de 25 MHz - la fiecare puls al acestuia se va trimite informația de culoare a unui anumit pixel) și produce următoarele semnale:

- HS - Sincronizare orizontală, acest semnal comunică dizpozitivului de afișare faptul că am terminat de afișat un rând și dorim trecerea fasciculului de electroni la începutul rândului următor;

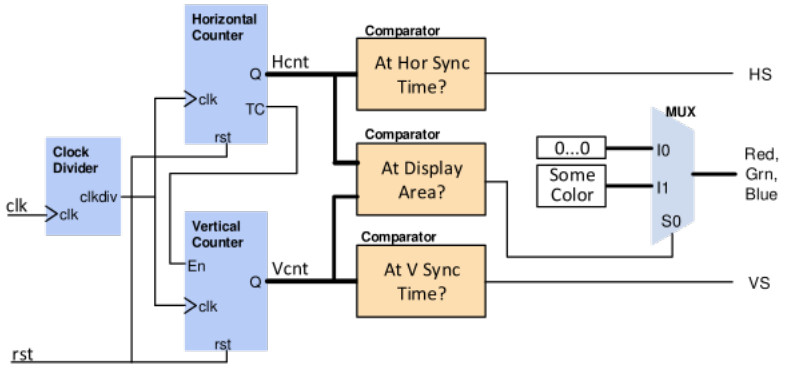

Fig. 7: Implementarea controller-ului VGA

- VS - Sincronizare verticală, acest semnal comunică dizpozitivului de afișare faptul că am terminat de afișat întregul cadru și dorim trecerea fasciculului de electroni în colțul stânga sus al ecranului, pentru afișarea unui cadru nou;
- H - Vector de zece biți, reprezintă poziția orizontală (coordonata X), a fasciculului de electroni, respectiv a pixelului curent;
- V - Vector de zece biți, reprezintă poziția verticală (coordonata Y), a fasciculului de electroni, respectiv a pixelului curent;
- Blank - Acest semnal este activ dacă valorile semnalelor H și V nu reprezintă poziții de pixeli valide. Între trecerile dintre linii sau cadre, este nevoie de timp suplimentar pentru sincronizare, în care informația de culoare nu va fi transmisă, ci se va trimite culoarea neagră / blank (valorile R, G și B vor fi zero), deoarece atunci monitorul va fi nevoit sa își calibreze culorile.

Semnalele H și V sunt folosite pentru a forma o adresă ce va indexa culoarea dintr-o memorie ROM ce conține o imagine, precum și pentru a determina dacă pixelul curent conține un obiect în modulul Snake.

Deși acest controller este capabil de a afișa imagini la o rezoluție de 640x480, datorită resurselor hardware (capacitatea de memorie a plăcii de dezvoltare FPGA) limitate, imaginile afișate sunt reduse la 160x120 (de 4 ori în fiecare dimensiune). În acest demers, numerele reprezentate de semnalele H și V au fost împărțite cu 4. Astfel, câte 16 combinații ale valorilor semnalelor H și V vor adresa aceeași locație de memorie. Pentru modulul Snake, se folosesc semnale H și V împărțite la 8, deci rezoluția jocului va fi 80x60.

Acest controller VGA a fost implementat utilizând două numărătoare ca module principale. Primul numărător va număra poziția orizontală a pixelului. După ce a terminat secvența de numărare, se va reseta la zero iar al doilea numărător va număra o singură dată. În Fig.7 este prezentată schema bloc a acestei implementări. Semnalul Blank inversat a fost aplicat la intrarea multiplexorului pentru a decide dacă se va transmite informația de culoare sau valoarea zero.

În Fig.8 este prezentată diagrama de timp a semnalelor VGA. Din aceasta

Descriere	Notație	Timp	Timp/Frecvența
Pixel Clock	t_{clk}	39.7 ns	25.175MHz
H Sync Time	t_{hs}	3.813 μs	96 Pixeli
H Back Porch	t_{hbp}	1.907 μs	48 Pixeli
H Front Porch	t_{hfp}	0.636 μs	16 Pixeli
H Addr Video Time	t_{haddr}	25.422 μs	640 Pixeli
H L/R Border	t_{hbd}	0 μs	0 Pixeli
V Sync Time	t_{vs}	0.064 ms	2 Linii
V Back Porch	t_{vbp}	1.048 ms	33 Linii
V Front Porch	t_{vfp}	0.318 ms	10 Linii
V Addr Video Time	t_{vaddr}	15.253 ms	480 Linii
V T/B Border	t_{vbd}	0 ms	0 Linii

Tabel 1: Specificația timpilor pentru controller-ul VGA

putem deduce următoarele intervale de timp:

$$t_{hs}, t_{hpb}, t_{hfp}, t_{haddr}, t_{hbd}, t_{vs}, t_{vbp}, t_{vfp}, t_{vaddr}, t_{vbd}$$

Aceste intervale sunt descrise în tabelul 1, împreună cu duratele lor. Vom afișa informație video doar atunci când numărătorul orizontal se află în t_{haddr} (are valoarea mai mică decât $H_{max_{addr}}$) iar numărătorul vertical se află în t_{vaddr} (are valoarea mai mică decât $V_{max_{addr}}$). În toate celelalte intervale de timp, informația de culoare va fi "blanked". Semnalele HS și VS vor fi active doar în timpii t_{hs}, respectiv t_{vs}.

Cu ajutorul specificației timpilor din tabelul 1, putem calcula valorile maxime ale numărătoarelor:

$$H_{max} = \frac{t_{hs} + t_{hbp} + t_{hfp} + t_{haddr} + t_{hbd}}{t_{clk}} = 800$$

$$V_{max} = \frac{t_{vs} + t_{vbp} + t_{vfp} + t_{vaddr} + t_{vbd}}{t_{hs} + t_{hbp} + t_{hfp} + t_{haddr} + t_{hbd}} = 525$$

$$H_{max_{addr}} = \frac{t_{haddr}}{t_{clk}} = 640$$

$$V_{max_{addr}} = \frac{t_{vaddr}}{t_{hs} + t_{hbp} + t_{hfp} + t_{haddr} + t_{hbd}} = 480$$

Pentru a calcula adresa pentru indexarea pixelului dintr-o memorie cu o imagine, se înmulțește V cu 160 (lungimea imaginii), iar apoi se adună cu H. Trebuie menționat faptul că se folosesc semnalele H și V împărțite la 4.

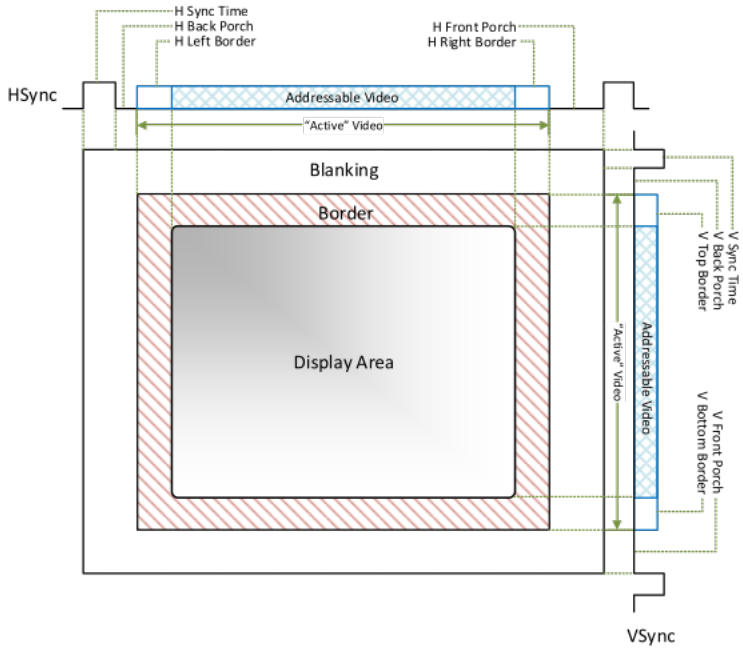

Fig. 8: Diagrama de timp pentru sincronizarea semnalelor VGA

4.2 Filtre de imagini

La implementarea filtrului de contract scăzut, în formulele din secțiunea 3.3.1, a fost ales:

$$F = \frac{1}{4}$$

astfel încat să se poată realiza împărțirea cât mai rapid, cu o deplasarea la dreapta cu doi biți. Similar, pentru filtrul de contrast ridicat, pentru a se putea realiza ușor înmulțirea cu o deplasare la stânga cu doi biți, s-a ales:

$$F = 4$$

Pentru a realiza conversia imaginii în grayscale, este nevoie de a executa operația de împărțire la 3, pentru obținerea medii aritmetice a valorilor cananelor fiecărui pixel. Această operație trebuie făcută rapid, astfel încât ar trebui să dureze mai puțin de 40 ns (un pixel trebuie trimis spre afișare la fiecare 25 MHz). Ne putem folosi de faptul că mereu împărțitorul operației este 3, și astfel am proiectat un circuit specializat ce realizeaza doar operații de împărțire a numerelor la 3. Circuitul obținut este unul iterativ, componentă necesară fiind un împărțitor de numere pe 4 biți la 3. Acesta a fost implementat ca un lookup table, luând în considerare numărul mic de combinații ale intrărilor - 16. Fiecare componentă va calcua un cât parțial ș, folosind restul parțial de la operația precedentă împreună cu 2 biți din deîmpărțit. La intrarea fiecărei componente vor fi aplicați câte 2 biți ai deîmpărțitului concatenați la stânga cu restul parțial de la componenta anterioară. Pentru prima componentă, considerăm restul anterior ca fiind zero.

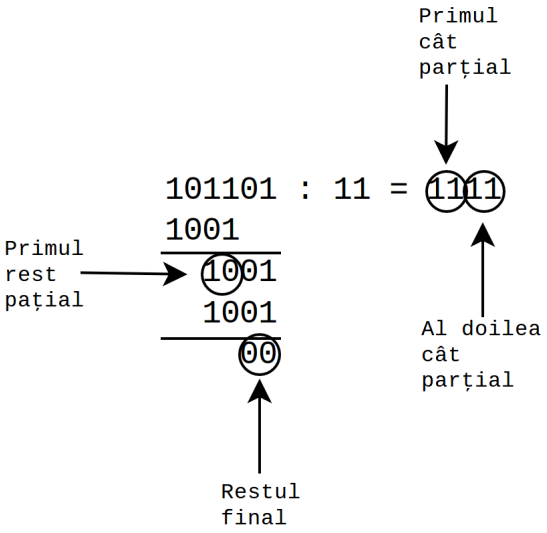

Fig. 9: Exemplificarea algoritmului de împărțire la 3

4.3 Modulul Snake

Jocul rulează la rezoluția de 80x60 pixeli. În jocul Snake există două obiecte: șarpele și mâncarea. Mâncarea este reprezentată cu un pixel. Aceasta își va schimba poziția de fiecare dată când intră în contact cu șarpele. Această poziție este memorată într-un registru de date. Șarpele este alcătuit dintr-un număr variabil de segmente. Acest număr crește de fiecare dată când capul (primul segment) intră în contact cu mâncarea. Când jocul începe, șarpele are patru segmente. Coordonatele segmentelor sunt memorate într-o serie de regiștri care se comportă ca o stivă. Alt registru este folosit pentru a memora direcția șarpelui. Există patru direcții posibile (sus, jos, stânga, dreapta), astfel 2 biți fiind suficienți. Valoarea din acest registru se va schimba la primirea comenzilor (de la tastatură, butoanele de pe placa de dezvolate). Într-un ultim registru este ținut scorul obținut. Inițial zero, acesta crește la fiecare coliziune a șarpelui cu mâncarea. Această valoare va fi afișată pe afișoarele cu 7 segmente.

În fiecare registru din stivă va fi reținut și un bit V, care determină dacă segmentul cu acea poziție face parte din șarpe. Avem nevoie de un astfel de bit deoarece numărul de segmente este variabil, în timp ce numărul de registre este fix. În fiecare cadru este calculată poziția următoare a capului șarpelui, în funcție de direcție. La momentul mișcării, pe stivă este dat push la noua poziție. Astfel, capul șarpelui fiind acum la poziția nou calculată, registrele următoare iau succesiv valorile registrelor anterioare. Bitul V al valorii nou introduse va fi 1. Totuși, valorile biților V își vor păstra valoarea, deoarece nu vrem ca numărul de segmente să crească. La momentul coliziunii cu mâncarea, deoarece dorim și creșterea numărului de segmente, vom deplasa inclusiv și valorile biților V. De asemenea, este folosit un generator de numere aleatoare pentru a genera o nouă poziție pentru mâncare și va fi incrementat scorul.

Generatorul de numere aleatoare a fost implementat cu algoritmul Xorshift.

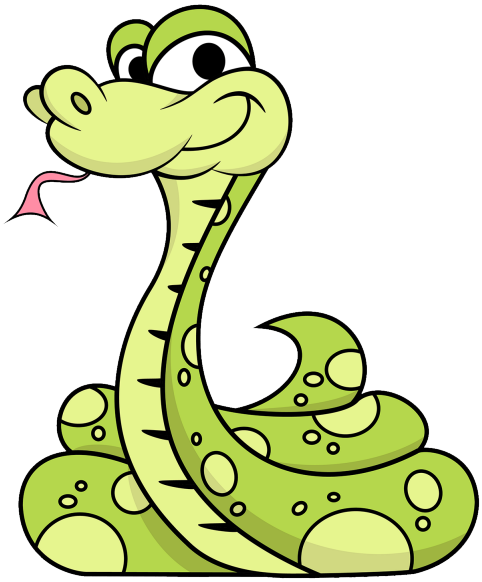

Fig. 10: Șarpe

Conversia scorului într-un număr BCD, pentru a fi afișat, este realizată cu algoritmul Double dabble.

Două intrări ale acestui modul, X și Y, coordonatele pixelului curent ale controllerului VGA, împărțite la 8, sunt comparate cu fiecare valoare din registrele stivă și cu valoarea din registrul pentru mâncare. Dacă există cel puțin o egalitate între coordonatele de la intrare și cele menționate anterior, iar, in cazul egalității cu o valoare dintr-un registru stivă, bitul V este 1, atunci ieșirea "Pixel On" va fi 1, iar în caz contrar 0. Această ieșire va determina dacă pixelul care urmează să fie desenat pe ecran conține un obiect din joc (șarpe sau mâncare). Culoarea albă reprezintă prezența unui obiect, în timp ce culoarea neagră absența acestuia. Această culoare va trece apoi prin filtrele de imagini.

Stiva a fost proiectată ca un modul generic, existând posibilitatea schimbării numărului de regiștri prin modificarea unei constante. Pentru implementarea curentă, această constantă a fost setată pe 256. Acest număr determină numărul maxim de segmente pe care îl poate avea șarpele. Deoarece acest număr determină și numărul de comparații cu coordonatele de la intrare, acesta nu trebuie să fie prea mare, pentru ca timpul de calcul să se încadreze, împreună cu celelalte calcule și întârzieri, în perioada semnalului "Pixel Clock".

Un număr suplimentar de comparatoare este necesar pentru a verifica coliziunea capului șarpelui cu restul corpului său. Dacă există o egalitate între coordonatele capului și una din coordonatele celorlalte segmente, iar bitul V al segmentului cu a cărui coordonate sunt egale este 1, atunci există coliziune. În caz că există coliziune, jocul se va termina. În acel moment, scorul va fi resetat la zero, poziția șarpelui va fi resetată la coordonatele inițiale, numărul de segmente la 4, iar mâncarea va primi o nouă poziție aleatoare.

Utilizare LUT	5377
Utilizare FF	3912
Utilizare BRAM	24
Utilizare IO	58
Utilizare BUFG	1
Frecvenţa maximă	129.6 MHz
Puterea totală	0.128 W
Temperatura	25.6 °C

Tabel 2: Informații din rapoartele de implementare

4.4 Citirea tastelor

Controller-ul PS/2 a fost realizat cu ajutorul unui registru de deplasare pe 11 biți. Pinul Clock al conectorului PS/2 este conectat la intrarea Clock al registrului, iar pinul Data la intrarea serială a acestuia. Un numărător numără biții recepționați. Atunci când a numărat 11 biți, se verifică dacă bitul de start, bitul de stop și bitul de paritate corespund specificației PS/2. Dacă acesta este cazul, este interpretat scan code-ul și se verifică dacă acesta corespunde uneia dintre tastele W, A, S, D. Starea acestor butoane (dacă este apăsat sau nu) este memorată într-un set de 4 bistabile. Dacă scan code-ul este un make code, atunci starea bistabilului va deveni 1, iar dacă este un break code, va deveni 0. Starea acestor taste, va fi folosită mai departe în modulul Snake.

5 Rezultate experimentale

5.1 Instrumente utilizate

Proiectul a fost implementat în limbajul de descriere hardware VHDL, în mediul de dezvoltare Xilinx Vivado 2017.4, pentru placa de dezvoltare Xilinx Basys 3 Artix-7 FPGA. Dezvoltarea proiectului a avut loc pe un laptop Asus F550J, cu sistemul de operare Microsoft Windows 10 64-bit. Alte instrumente utilizate la dezvoltare sunt un monitor LG 24MP68VQ-P și o tastatură Cooler Master STORM QuickFire XT MX Brown.

5.2 Circuitul implementat

În tabelul 2 sunt prezentate informațiile din rapoartele de implementare oferite de mediul de dezvoltare Vivado.

Calea critică a întregului circuit este reprezentată de comparatoarele din modulul Snake (secțiunea 4.3). Dacă creștem numărul registrelor din stivă și, implicit, numărul comparatoarelor, vom observa o scădere a frecvenței maxime. Totuși, valoarea obținută până acum este destul de departe de valoarea minimă de care avem nevoie (25 MHz), deci s-ar putea experimenta cu un număr mai mare de registre.

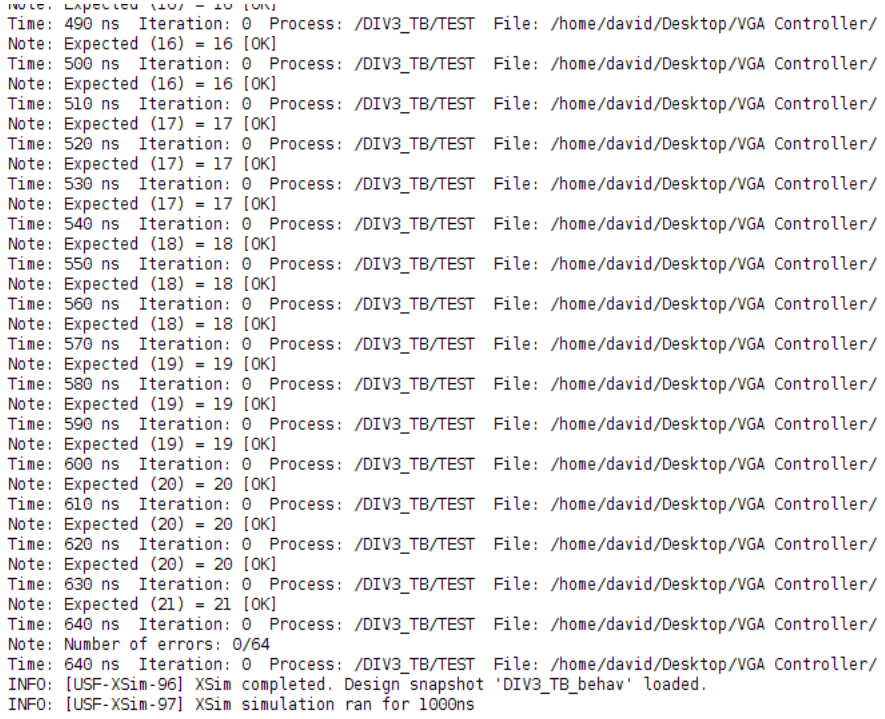

Fig. 11: Rezultatele simulării împărțitorului la 3

5.3 Testare

Pentru testarea sistemului, a fost utilizat simulatorul Vivado. A fost scris un banc de test pentru împărțitorul la 3, în care a fost aplicat un număr de 64 de vectori de test. Testele au fost trecute cu succes, după cum se poate observa în Fig. 11.

6 Concluzii

Problema rezolvată a fost aceea a intefațării conectorului VGA, în vederea afișării imaginilor pe un display cu conectivitate VGA. Obiectivul inițial a fost acela a afișării unei imagini statice color pe un display. Scrierea imaginii într-o memorie ROM ar fi fost o sarcină lungă și obisitoare. Pentru a rezolva această problemă, a fost dezvoltată o aplicație în C++ cu librăria OpenCV, care primește o imagine la intrare și generează un fișier sursă VHDL cu memoria ROM ce conține culorile pixelilor imaginii. Pentru a putea stoca o imagine la rezoluția nativă (640x480), Block RAM-ul plăcii de dezvoltare nu ar fi fost suficient, astfel imaginea a fost redusă ca dimensiune de către aplicația menționată anterior și s-a efectuat o operație de scalare pe coordonatele pixelilor. Un ultim pas de procesare a imaginii înainte de scrierea memoriei a fost reducerea gamei de valori a fiecărui canal de culoare de la 0-255 la 0-15, printr-o deplasare la dreapta cu 4 biți, deoarece rezoluția convertorului digital-analog al plăcii de dezvoltare - ce este utilizat pentru a converti canalele de culoare în analog pentru a le putea transmite prin pinii vgaRed, vga-

Green și vgaBlue către display - este de 4 biți. Acest obiectiv fiind îndeplinit într-un timp relativ scurt și, neexistând constrângeri de timp, au mai fost stabilite alte trei obiective:

- Aplicarea unor filtre de imagini prin modificarea valorilor cananelor de culoare în timpul execuției. A fost necesar ca filtrele să fie componente combinaționale mici care vor executa calcule rapid, pentru a nu introduce întârzieri semnificative. O provocare a fost filtrul grayscale, care necesită o împărțire la 3. În secțiunea 4.2 este prezentat algoritmul de împărțire.
- Afișarea de imagini dinamice, interactive, care răspund la stimulii utilizatorului. Acest obiectiv s-a concretizat prin dezvoltarea unui joc 2D simplu, mai exact o implementare a jocului Snake. Acest gen de joc a avut debutul în anul 1976, pe hardware-ul de jocuri arcade Blockade [5]. Astfel, datorită trivialității implementare și a resurselor hardware necesare reduse, a fost aleasă această aplicație. O dificultate a reprezentat-o posibilitatea ca șarpele să se poată întoarce direct, în direcția opusă, la 180°. O astfel de mutare avea drept consecință imediată sfârșitul jocului. A trebuit implementată o restricționare a mișcărilor posibile, prin adăugarea de logică suplimentară la intrarea Write Enable a registrului de direcție. Datorită numărului mare de registre și porți logice, a fost aleasă o implementare de tip for...generate, pentru a reduce dimensiunea codului, pentru a crește simplitatea și lizibilitatea, dar și pentru a putea aduce modificări cu ușurință. Pe parcursul dezvoltării, a fost modificat în mod repetat numărul de registre din stiva șarpelui, testând de fiecare dată funcționalitatea corectă a modulului.
- Preluarea de input de la o tastatură PS/2 pentru a intercțiunea cu utilizatorul. Au existat anumite dificultăți legate de sincronizarea semnalelor, ce au fost rezolvate prin experimentarea cu diferite întârzieri introduse artificial.

6.1 Dezvoltări viitoare

O posibilă dezvoltare ulterioară ar putea fi adăugarea unui microprocesor cu intrucțiuni speciale de manipulare a unei memorii video (framebuffer). Controller-ul VGA va afișa pe display informațiile din acest framebuffer. Mai multe jocuri ar putea fi scrise în cod mașină pentru acest microprocesor (Pong, Asteroids, Tetris, Breakout). Astfel, acest proiect ar deveni o platformă de jocuri de tipul jocurilor arcade din anii 1970-1980.

Alte posibile dezvoltări, dar care ar putea necesita o placă de dezvoltare cu mai multă putere de procesare:

- Exemplificare de randare 3D cu un ray tracer;
- Video streaming de pe rețea. Este necesar un port de ethernet și ieșire audio;
- Afișarea de date despre stock market sau piața criptomonedelor sub formă de grafice și tabele. Preluarea datelor se va face de la un server;
- Display interactiv pentru a afișa informații despre locație sau împrejurimi într-un loc public, cum ar fi un mall, o grădină zoo, un muzeu, sau alte locuri vizitate de turiști;
- Vizualizator sub formă de unde luminoase pentru muzică.

Referințe

[1] Adam Chapweske. The ps/2 mouse/keyboard protocol, 2003.

[2] Digilent Inc. Vga display controller, 2014.

[3] Francis G. Loch. Image processing algorithms part 5: Contrast adjustment. *The Crypt*, 56, 2015.

[4] Sergiu Nedevschi, Tiberiu Marita, Radu Danescu, Florin Oniga, Raluca Brehar, Ion Giosan, and Cristian Vicas. *Procesarea imaginilor*. U.T. PRESS, 2013.

[5] Wikipedia contributors. Snake (video game genre) — Wikipedia, the free encyclopedia, 2018.

[6] Wikipedia contributors. Video graphics array — Wikipedia, the free encyclopedia, 2018.