UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Catedra de Calculatoare şi Tehnologia Informaţiei

DOCUMENTAŢIE

Structura Sistemelor de Calcul

Proiect: Afişarea unei imagini folosind portul VGA

Nume Student: Boloş Andrei-Nicolae
Grupa: 30236
Îndrumător Proiect: Prof. Şl. Dr. Ing. Dragoş Florin Lişman

CUPRINS

1. Rezumat

Obiectivul principal al aplicaţiei reprezintă proiectarea şi implementarea unui modul VHDL care să permită afişarea unor imagini stocate in memoria plăcuţei pe un monitor, folosind portul VGA.

2. Introducere

Tema proiectului constă în afişarea unei/unor imagini stocate in memorie pe un monitor folosind portul VGA al plăcuţei programate.

Acest proiect a fost implementat în mare parte în limbaj VHDL, folosindu-se de ajutorul unui cod scris in Matlab pentru conversia unei imagini în format text.

Limbajul VHDL este un limbaj de descriere a componentelor Hardware referitor la componenţa acestora şi interconexiunile dintre ele. VHDL a devenit un limbaj industrial standardizat.

Plăcuţa utilizată drept suport pentru acest proiect a fost una de tip FPGA *(Field Programmable Gate Array)* Basys-3 Artix-7. Aceasta a fost programată folosind limbaj VHDL.

Soluţia propusă este alcătuită dintr-un modul de decizie ce permite utilizarea controlată a portului VGA *(Video Graphics Array)*, o memorie ROM în care este stocată informaţia corespunzătoare culorii fiecărui pixel al imaginii şi o componentă ce selectează 4 biţi din cei 8 aferenţi fiecărei culori *(RGB)*.

3. Fundamentare Teoretică

Principiul de funcţionare al afişării unei imagini constă în interpretarea ecranului precum o matrice de pixeli. Următoarele exemplificări vor avea rezoluţia imaginii de 640x480 pixeli. Informaţia de culoare a fiecărui pixel va fi stocată într-o memorie ROM de forma unei matrici de 640x480, unde fiecare câmp conţine câte 24 de biţi, adică 8 pentru fiecare culoare.

Controlul interfaţei VGA se rezumă la validarea semnalelor de sincronizare. Afişarea autentică se făcea pe ecranele CRT (Cathod Ray Tube), pe ale căror suprafaţă "se plimba" un electron pe fiecare linie de sus în jos, de la stânga la dreapta, formând "pixeli". Pixelii de la începutul ecranului (colţul stânga-sus) trebuie să rămână afişaţi până la terminarea interogării matricii de memorie ROM (colţul dreapta-jos).

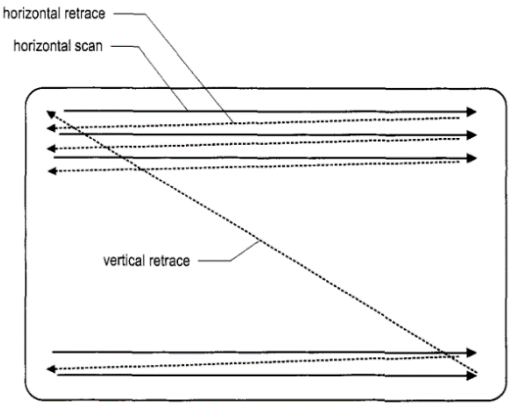


Figure 12.2 CRT scanning pattern.

Citirea matricii se face linie cu linie de la stânga la dreapta, de sus în jos. După citirea

primei linii, pointerul trebuie să revina de la finalul acesteia la începutul celei de-a doua linii *(retrace)*, după care urmează procesarea ei şi tot aşa mai departe până se ajunge la ultimul pixel de pe ultima linie. Atunci pointerul trebuie să revină la începutul primei linii pentru a lua de la capăt scanarea. La o rată de refresh a ecranului de 60 Hz, această scanare avea loc o la fiecare 1/60 secunde, rezultând o frecvenţă de procesare a pixelilor de 25MHz.

Semnalul *Horizontal Sync* are rolul de a marca finalizarea părţii de "retrace". Acest lucru se realizează prin trecerea lui din starea *HIGH* în stare *LOW* şi apoi iar *HIGH*. Decalajul de timp după această trecere se numeşte *Back Porch*. Apoi se trasează liniar rândul de pixeli. După această trasare, are loc un decalaj numit *Front Porch* înainte de schimbarea stării semnalului *Horizontal Sync*. Apoi tot acest algoritm o ia de la capăt odată cu începerea fiecărui rând nou.

Semnalul *Vertical Sync* marchează finalizarea trasării unui cadru întreg, adică o imagine.

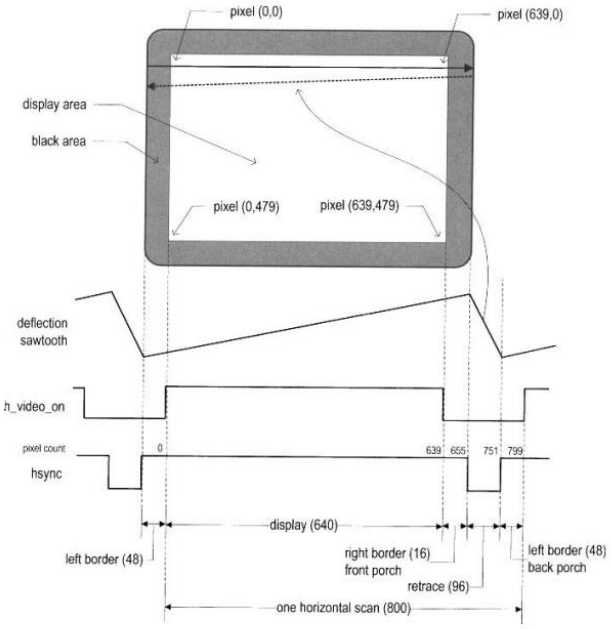
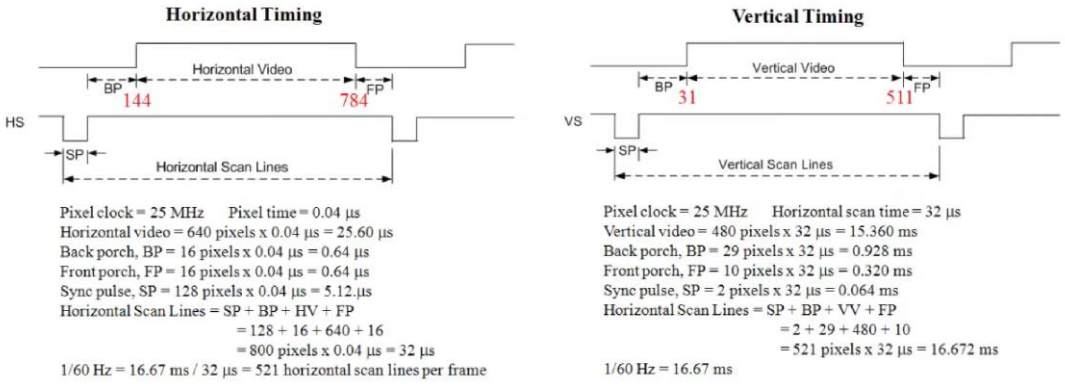
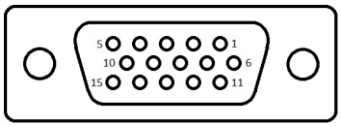


Figure 12.4 Timing diagram of a horizontal scan.



Portul VGA are la bază semnalele de sincronizare şi semnalele de culori (RGB).



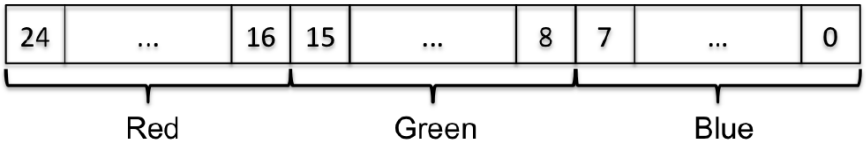
Imaginile folosite au adâncimea culorii de 24 de biţi. Asta înseamnă că fiecare culoare din cele 3 are alocată o zonă de câte 8 biţi.

Portul VGA al plăcuţei Basys-3 suportă doar 4 biţi pentru fiecare culoare.

Cum limbajul VHDL nu suportă lucrul cu imagini în formatul original (jpg/jpeg/asemănător), imaginea a trebuit convertită într-un fişier de tip .txt pentru a putea fi interpretată de limbajul VHDL.

Pentru realizarea aceste-i operaţii s-a utilizat Matlab. Codul sursă a fost scris de nişte studenţi. https://github.com/oleveque/VGA-Display/blob/master/src/gene_img.m

Având structura X"1A2B3C" pentru fiecare pixel, această formă oferă posibilitatea ca imaginea să fie stocată într-o memorie ROM.



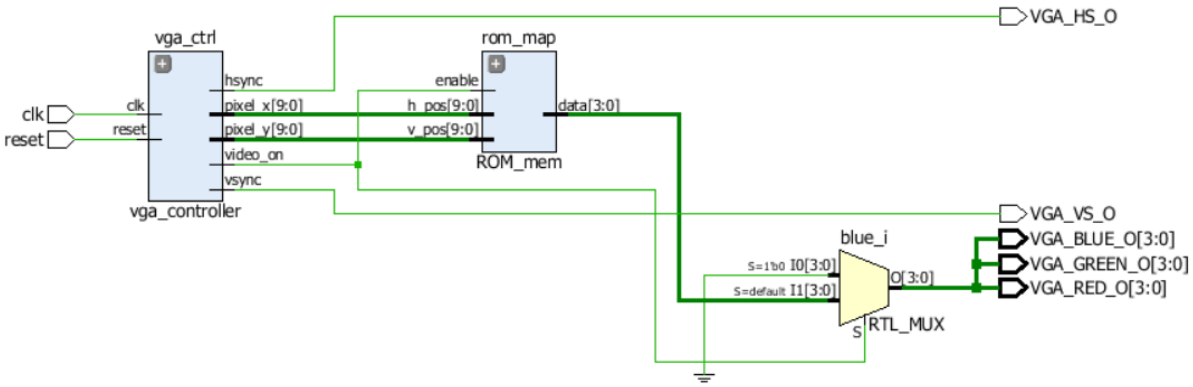
Abordarea firească este să mapăm aceste valori de culoare la semnalele de ieşire ale portului VGA. Pentru fiecare pixel sunt generaţi câte 24 de biţi ce reprezintă culoarea. Făcând înmulţirile aferente, 640x480*24, doar memoria ROM in sine are de lucru cu 7.372.800 biţi de memorie. Prin testare, mi-am dat seama că durează foarte mult pentru a simula şi a încărca pe plăcuţă.

De aceea am decis să optimizez numărul de biţi de lucru. Pentru un pixel, portul VGA necesită 12 biţi. Îmbunătăţirea adusă se rezumă la constrângerea imaginilor să fie afişate alb-negru. Aceste imagini alb-negru au proprietatea că valorile biţilor pentru cele 3 culori sunt egale (ex: x”2A2A2A”). Fructificând această informaţie, am decis să salvez doar 4 biţi în loc de 12, ceea ce reduce mărimea memoriei ROM alocate de circa 3 ori. Astfel reducându-se numărul de biţi de lucru de la 7.3 mil. la doar 2.4 mil. Din testare, într-adevăr “se simte” diferenţa în timpul de aşteptare necesar generării sintezei şi a implementării.

Ca finalizare a soluţiei, se face asignarea semnalelor obţinute din parsarea imaginii şi cele de sincronizare la semnalele de output ale portului VGA.

4. Proiectare și Implementare

Schema Block:



Pentru sincronizare s-a implementat un modul *(vga_controller.vhd)*. Acest modul este implementat pe baza logicii explicate în cartea atașată în bibliografie, la paginile 260-266 [1] și pe baza unui tutorial de pe youtube [2]. Se creează un divizor de frecvență pentru a reduce viteza de accesare a pixelilor, de la cei 100 de MHz inițiali ai plăcuței, la 25 MHz, folosing un numărator pe 2 biți cu ajutorul căreia se determină schimbarea celui mai semnificativ bit *(primul)* din '0' în '1'. Se folosesc număratoare *(mod 525, mod 800)* pentru a determina finalul zonelor delimitate pentru sincronizare. Se folosește un semnal de enable pentru a testa dacă pixelul se află în zona corespunzătoare afișării sau nu.

Pentru stocarea imaginii s-a creat un modul de memorie *(ROM_mem.vhd)*, care are la bază o matrice de 640x480 de câmpuri, fiecare conținând 4 biți de valoare pentru culoare/nuanță de gri *(cei mai semnificativi din cei 12 biți de culoare)*. Cum în exemplul demonstrativ am selectat o imagine alb-negru, am ales să implementez o variantă "lite" a soluției. La selectarea celulei de memorie s-a testat *(în vga_display.vhd)* dacă locația acesteia în matrice și în ecran este validă, adică nu aparține unei zone rezervate pentru sincronizare *(front/back porch)* sau dacă este în afara zonei acoperită de rezoluție.

Componenta principală *(vga_display.vhd)* face conexiunea dintre componenta de control pentru sincronizare și componenta care selectează biții pentru imagine din memoria ROM. Această componentă stabilește semnalele de output.

5. Rezultate experimentale

Partea de testare reprezintă verificarea funcţionalităţii aplicaţiei în timp ce aceasta se dezvolta şi după ce a fost finalizată.

6. Concluzii

Acest proiect a avut efectul de a mă învăţa cum se făcea afişarea pe la începutul folosirii în masă a monitoarelor CRT şi a porturilor VGA.

7. Bibliografie

1. Pong P. Chu, “FPGA Prototyping by VHDL Examples, Xillinx Spartan-3 Version”, Cleveland State University, 2008, by John Wiley & Sons, Inc, https://blog.aku.edu.tr/ismailkoyuncu/files/2017/04/02_ebook.pdf
2. https://www.youtube.com/watch?v=eJMYVLPX0no&list=LL&index=3&t=324s&ab_channel=daxerz
3. https://www.youtube.com/watch?v=wzhDRIX2Ors&ab_channel=LBEbooks
4. https://www.youtube.com/watch?v=7j7brGz7u6M&ab_channel=LBEbooks