Binary Love

The story of a lonely stick figure in a FPGA world.

Carla Gerst, Xiangyuan Guo, Kelley Harris, Xinyang Jian
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mail's: carlagerst@oakland.edu, xiangyuanguo@oakland.edu, kjharri2@oakland.edu, jian@oakland.edu

The purpose of this final project and presentation was to incorporate some important aspects that were covered throughout this digital logic design course, while also maintaining some skills developed in previous engineering courses. Using the Xilinx university program, this project was implemented using the Video Graphics Array (VGA) controller on the Nexys4 DDR Field Programmable Gate Array (FPGA). An image was uploaded from MATLAB and downloaded onto the VGA controller of the FPGA board. This procedure allowed for the understanding of how the VGA controller works and gave a glimpse into what can be done to an image displayed on a monitor.

I. Introduction

The purpose of this project was to create a fun and interactive program that allowed the user to cover certain pixel ranges of an image that would be hidden during four different states. This image was displayed on a monitor while a story full of vocabulary relating to this class was told. This original image was created using Adobe Photoshop, and converted into a text file using MATLAB. This test file was then downloaded onto the FPGA board and displayed on the monitor using the VGA controller. The RGB switches were enabled so that the background color could also be manually changed by the use of various combinations of the 12 rightmost switches which use a specific pattern to create a range of nine colors.

II. Methodology

The motivation for the project stems from the many different things that were taught in the class that can be done with the Nexys4 DDR FPGA board. In an attempt to keep the audience engaged, a creative tale of two binary bits were used and the story let the audience feel how lonely it could be being one single bit in such a large array of seemingly endless options.

A. Overview

To tell this exciting tale, an image was created using Adobe Photoshop Elements 9, (Figure 1). This original image was then downloaded using MATLAB. MATLAB was used to convert the image into a useable text file for importing onto the FPGA board. The VGA control program that was provided on the class website [1] was used and slightly modified to fit the details of this story. A Finite State Machine (FSM) was created following along with the algorithmic state diagram, (Figure 5) and the data path circuit, (Figure 3). The program was implemented onto the FPGA board and the image was projected onto a monitor for presentation.

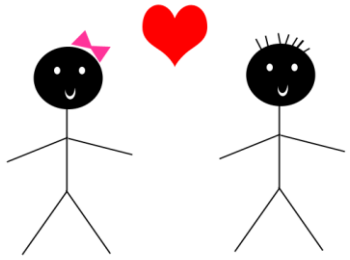

Figure 1. Binary Love Birds

B. MATLAB

To begin, a simple image of two stick figures and a heart was created using Adobe Photoshop Elements 9. This image was then read into the MATLAB file. The image was resized to fit in a 256x256 pixel range and converted to a 12-bit RGB image. This would allow the image to be easily downloaded onto the FPGA board. The MATLAB code that was used was taken from the class website [1] and only slightly edited to read the new image file while eliminating the default android image that was used for the class website's example. The exact code that was used is located in below, (Figure 2).

```
clear all; close all; clc;

I = imread ('binarylove.png'); % RGB image
figure;imshow(I);

% Resizing the image to 256x256:
IP = imresize(I,[256 256]);
figure; imshow (IP);

% 24-bit RGB image: we will convert it to a 12-bit RGB image:
for i = 1:3
    IN(:,:,i) = IP(:,:,i)/16; % every plane converted to 4 bits. right shift
end

figure; imshow(IN*16); % This is just so that 'imshow' can display the image properly

% ------------------------------------------------------------------------
% Converting to text file. Format: 0|R|G|B in hexadecimal
q = quantizer ('ufixed', 'round', 'saturate', [4 0]);
textfile = 'myimg.txt';
fid = fopen (textfile, 'wt'); % generates text file in write mode

for i = 1:256
    for j = 1:256
        R = IN(i,j,1); G = IN(i,j,2); B = IN(i,j,3);
        Rh = num2hex(q, double(R)); Gh = num2hex(q, double(G)); Bh = num2hex(q, double(B));
        fprintf(fid, '0%s%s%s\n',Rh, Gh, Bh);
    end
end

fclose (fid);
```

Figure 2: MATLAB code

C. VGA Controller

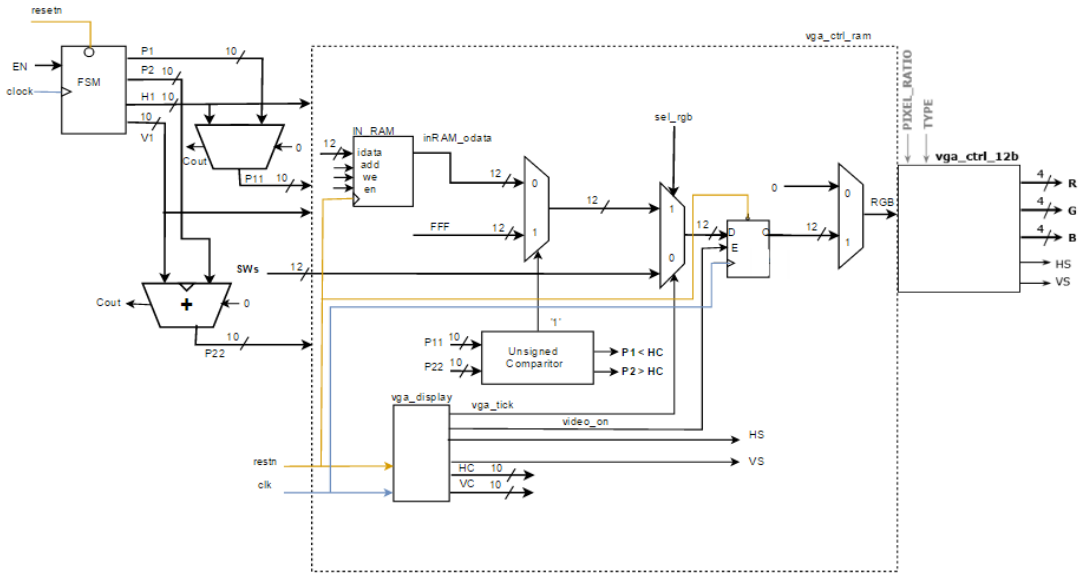

Figure 3: Data path Circuit

Once the MATLAB text file was created, the image could then be used and implemented using the VGA controller of the FPGA board. The program code for the VGA controller was also the provided on the class website [1]. This available code was modified specifically for the different masks used in the different states and allowed for the image to move to different areas of the monitor. The modified portion of the code is located below, (Figure 11). The diagram in Figure 5, the Algorithmic State Machine (ASM) was used to control the parameters for the mask and the shifting parameters for the image. Figure 3 is the circuit diagram that was completed to help with creating the correct program and was also used to visually show how the circuit was programmed in Xilinx using a Finite State Machine.

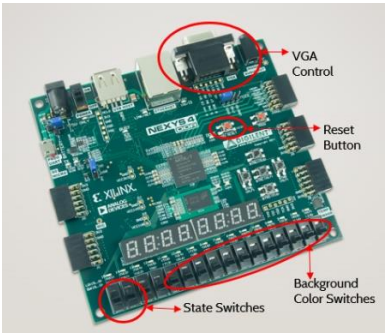

Figure 4: FPGA NEXYS4 DDR

D. Finite State Machine

A Finite State Machine (FSM) was created using the Xilinx university program. This followed along with the algorithmic state diagram, (Figure 5) and the data path circuit diagram, (Figure 3). Four states were created to indicate what part of the story was to be told. The states began with the last two switches, (V10 & U11), on the FPGA board in a low output with a binary number of 00.

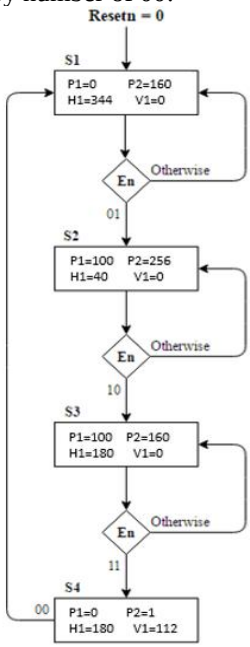

Figure 5: Algorithmic State Machine

The FSM was set to initiate only once the restartn was activated as a high output. From here the FSM moved into its first state and the story began.

During state one, the pixels of the girl stick figure and the red heart were covered, or masked, exposing the pixels that included only that of the male stick figure. The image used in state one is located below, (Figure 6). The actual image from state one included the entire image but displayed only the pixels relating to the boy stick figure. The pixels that include the girl and the heart were masked using the state machine, so the output only included what was mentioned in the story at that point.

In state one, the enable worked only if the last two switches on the FPGA read the output as a binary number of 1. This meant that the switches used to control the image were indicated as; V10 in the low position, and U22 was in the high position. If the enable was not activated, then the state machine would stay in the first state. If the enable was activated, or when the binary output number on the switches equaled 2, (V10-high, U11-low), the FSM would move onto the next state. The image shown in Figure 6 shows the specifics of what was shown on the monitor and what was hidden behind the mask. The mask is shown translucent in order to show the mask coverage area. In practice, it would be completely which to the girl stick figure and heart would not be detectable.

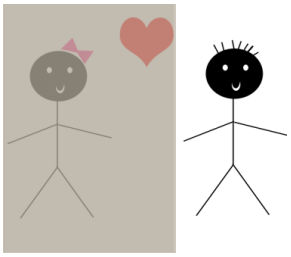

Figure 6: State one's image

Once the enable was activated and the program reached the next state, state two, the mask was relocated to cover the pixels of the male stick figure along with the heart. This pixel range allowed the image to expose the pixels that were only related to the girl stick figure (Figure7). This was done using the last two switches on the FPGA board when the binary output was 2, or the switches, V10 was low and U11 was high. If the enable was not activated, or when both switches were an active low, than the machine would stay at state two. If the enable was activated, then the program could proceed to move towards the next state.

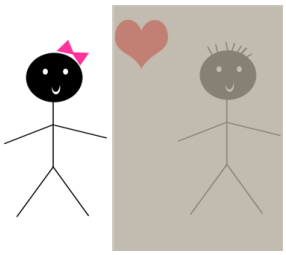

Figure 7: State Two's image

If the enable from state two was activated, then the program would continue onto state three. At this state in the love story, the two lonely stick figure bits finally meet as shown in (Figure 8). In this figure, the pixels of the two stick figures are visible on the monitor leaving only the pixels of the red heart covered. This is only enabled when the binary number of the last two switches equals 3, or when both switches, V10 and U11 are at an active high output. If the enable is not activated, then the image stays in state three. If the enable is activated, then the FSM continues and moves onto fourth and final state.

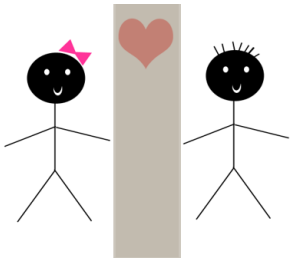

Figure 8: State Three

In the fourth state, the entire pixel range of the image was finally visible on the monitor, thus exposing the pixels of the red heart along with both stick figures, (Figure 1). During the story, this final state symbolized that and two lonely bits have finally fallen in love and are no longer two separate lonely bits in an FPGA world. Once this final state is complete, the FSM can then move back to the first state and the love story can begin once again.

If at any time, the user would like to start the story over, the reset button located on the FPGA board, (Figure 4) can be pressed and the program will restart at state one.

The Finite State Machine code that was used is located in Figure 9 and the modified parts of relating to the provided VGA code that was used in this project are located in Figure 10.

```
31  architecture Behavioral of FSM_cover is
32
33      type state is (S1,S2,S3,S4);
34      signal y: state;
35
36
37  begin
38
39      Transitions: process (resetn, clock, en)
40      begin
41          if resetn = '0' then -- asynchronous signal
42              y <= S1; -- if resetn asserted, go to initial state: S1
43          elsif (clock'event and clock = '1') then
44              case y is
45                  when S1 =>
46                      if en = "01" then y<=S2; else y<=S1; end if;
47                  when S2 =>
48                      if en = "10" then y<=S3; else y<=S2; end if;
49                  when S3 =>
50                      if en = "11" then y<=S4; else y<=S3; end if;
51                  when S4 =>
52                      if en = "00" then y<=S1; else y<=S4; end if;
53              end case;
54          end if;
55      end process;
56
57      Outputs: process (y)
58      begin
59
60          case y is
61              when S1 => p1 <= "0000000000"; p2 <= "0010100000";  h1 <= "0101011000"; v1 <= "0000000000"; -- 0/160
62              when S2 => p1 <= "0001100100"; p2 <= "0100000000";  h1 <= "0000101000"; v1 <= "0000000000"; --100/256
63              when S3 => p1 <= "0001100100"; p2 <= "0010100000";  h1 <= "0010111000"; v1 <= "0000000000"; --100/160
64              when S4 => p1 <= "0000000000"; p2 <= "0000000001";  h1 <= "0010111000"; v1 <= "0001110000"; --none
65          end case;
66      end process;
```

Figure 9: Finite State Machine Code

```
101  iram: in_RAM generic map (NDATA => NDATA, FILE_IMG => FILE_IMG)
102        port map (clock, resetn, inRAM_idata, inRAM_add, inRAM_we, inRAM_en, inRAM_odata);
103
104        -- Writing to this RAM is disabled here, but we can always enable to load new data
105        inRAM_idata <= (others => '0'); inRAM_en <= '1'; inRAM_we <= '0';
106
107        -- The following formulas only works if the image is square and the width size is a power of 2.
108
109
110        inRAM_int <= (conv_integer(vcount_buf)-conv_integer(v1))*256+ (conv_integer(hcount_buf)-conv_integer(h1));
111        inRAM_add <=conv_std_logic_vector(inRAM_int, 16);
112
113        -- This is how we control what appears on the screen (only a square 2**NBSP x 2**NBSP)
114        sel_RGB <= '1' when (h1 < hcount_buf and hcount_buf < 2**NBSP+h1) and (v1< vcount_buf and vcount_buf < 2**NBSP+v1) else '0';
115
116        in_RGB <= X"FFF" when (p1 < hcount_buf and hcount_buf < p2) else inRAM_odata (11 downto 0);
117
118        with sel_RGB select
119          | in_RGB2 <= in_RGB when '1', -- only the 12 LSBs contain useful data
120                       sw when others;
```

Figure 10: Changes to provided VGA code

III. Experimental Setup

This project was created using a PC computer and utilized the use of the Nexys4 DDR FPGA board's VGA controller, along with the programs: Xilinx University Program, MATLAB, and Adobe Photoshop Elements 9. A VGA cable was connected from the board to a monitor where the image was projected. The two left most switches on the FPGA board, V10 and U11, manually controlled the image while the right most 12 switches, (J15, L16, M13, R15, R17, T18, V18, R13, T8, U8, R16, and T13), (Figure 4), controlled which color could be display in the background behind the image.

The expectation of this project would be considered successful if, in state one, the imported image would appear in the upper right corner of the monitor. With the visible pixel ranges exposing the boy stick figure. In state two, the image would appear in the monitor's upper left corner. The visible pixel ranges exposing only the girl stick figure. In state three, the image would appear in the top middle of the monitor. With only the two stick figures visible. Then, in state four, the entire image would appear, unmasked, the below where state three appeared but near the bottom of the monitor screen.

IV. Results

This program was implemented by using the techniques explained above. Therefore, by adjusting the switches that controlled the state machine, a picture was obtained and appeared on a monitor using four different states. Even though switches that controlled the background color were initiated in the program's code, during the presentation, the background color stayed white and the switches remained in the active high position. These specific switches could be used to improve the program at a later date.

The specific pixels that were used as a mask started in state one with the pixel range from 0 to 160. The pixels in this range were part of the image that was covered by the mask. These numbers were found only after experimenting with the different pixel ranges until the correct combination was obtained. This range allowed for only the boy stick figure to be shown in the window, (Figure 6). For state two, the pixel range that was found to be correct was between 100 and 256. This range allowed for the visible window to show only the girl stick figure, (Figure 7). In state three, the pixel range 100 to 160 was used to cover up the heart in the middle of the image and exposing the boy and girl stick figures, (Figure 8). Finally, in state four, the pixel range that was used was from 0>P1 and P2<1 pixels. Since this condition is never true, no pixels are masked in this state. This specific range was used to eliminate any confusion that may have occurred in the program if the pixel range was equal to zero. This allowed for the image to appear in full on the monitor, (Figure 1).

After the correct pixel ranges were found, it was decided to have the image be displayed on different areas on the monitor. So here the vertical and horizontal values change specifically to where the image was placed on the screen. In state one the image appeared at a horizontal distance (H1) at 344 and the vertical distance (V1) at zero. This placed the image in the upper right corner of the screen. For state two, the vertical distance remained at zero, but the horizontal distance moved to the left side of the monitor with H1 equal to 40. This displayed the image in the left corner of the monitor. For state three the image was moved horizontally to 180 but stayed in the vertical location of zero. This displayed the image in the middle if the monitor at the top of the screen. Finally, for state four, the entire image was moved vertically and horizontally with H1 at 180 and V1 at 112. This placed the picture in the middle of the screen at the bottom of the monitor. Again, these numbers were found only after experimenting with different values until the desired positions were obtained.

A couple of issues were encountered while creating this program. The main issue that was encountered was that while the window and mask moved to different areas of the screen, the image did not move with it. Depending on the window location, it would only show part of an image. It was determined that the "inRAM_add" signal had not been modified. This was corrected shifting the HC and VC in the equation located in (Figure 10, line 110). Once this issue was corrected, the program ran smoothly. This project has given a great experience in implementing certain components learned throughout the course and programming them to work together for a common purpose.

V. Conclusions

This project gave a glimpse into what can be done with the Nexys4 DDR FPGA board. In summary, the group agreed, a lot was learned regarding what the board can do with images converted into text format, how to mask certain pixels of the image and how to project that image onto a monitor using the VGA controller. Also, this project showed the importance of creating the circuit diagram to serve as a guide for implementing the code in the Xilinx software. The program can be expanded and improved using the background color switches that were initialized, but not used for this particular story. This project was a starting point for communicating the "Binary Love" story but it could be further developed by adding different images, and states, to tell a longer or more complicated story.

Overall, this project was considered a successful example of some of the topics that were covered throughout this digital logic design course.

REFERENCES

[1] Dr. Daniel Llamocca, VHDL Coding for FPGAs---VGA Display Control
http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html