Sam Lee

CPE 166 – 03

Advanced Logic Design Lab

Wednesday: 5PM - 7:50 PM

Lecture Professor: Jing Pang

Lab Professor: Eric Carmi

Lab #5 Report

Contents

[Introduction 3](#_gjdgxs)

[Part 1&2: Microprocessor Data Path Design 3](#_30j0zll)

[Design Purpose 3](#_1fob9te)

[Engineering Data 3](#_3znysh7)

[Source Code](#_2et92p0) 5

[User Constraints](#_tyjcwt) 15

[Results Discussion](#_17dp8vu) 16

[Conclusion](#_3rdcrjn) 24

# Introduction:

# The purpose of lab 5 is to learn how to output a VGA Signal, through outputting colors and outputting the first letter of our name. We are doing this lab to introduce the concepts of how to generate VGA sync & timing signals, how to generate display colors on VGA and how to display patterns on VGA. These concepts are important because many electronics depend on video output. Throughout this lab I hope to learn how to use the VGA output and how

# improve my skills on FPGA programming in general.

# Part 1&2: Microprocessor Data Path Design

## Design Purpose:

The purpose of part 1 is to output different colors by manipulating the Red,

Green and Blue pins in VGA. The purpose of part 2 is to manipulate elements

within the 640x480 VGA frame to display things we want with this case being

the first letter of my name, S.

## Engineering Data:

## VGA is an analog video display standard. A standard VGA signal contains 640 columns by 480 rows of pixels. By turning on and off individual pixels, an image can be displayed on the monitor screen. The VGA signal scans left to right top to bottom and all signals need to be synced according to VESA standard for the display to take the signal.

## A VGA video signal contains 5 active signals:

## horizontal sync: digital signal, used for synchronization of the video

## vertical sync: digital signal, used for synchronization of the video

## red (R): analog signal used to control the color

## green (G): analog signal used to control the color

## blue (B): analog signal used to control the color

## The RGB signals control the pixel color at a given location, the HS signal determines timing for the row and VS determines timing to scan the entire screen. We need to manipulate them to output a picture in the PC Display. For Part 1 we change the color of the general VGA signal, and for Part 2 we draw a character with white pixels by manipulating the RGB Signals at specific

## points.

## Figure 1: This figure is the 640x480 resolution VGA mode horizontal timing.

## 

## Figure 2: This figure is the 640x480 resolution VGA mode vertical timing.

## 

## Figure 3: This figure shows the VGA port and FPGA pin connections.

## 

## Source Code:

## Color & Letter Verilog Code:

## module vga\_char (

## input wire clk,

## input wire clr,

## output wire hsync,

## output wire vsync,

## output reg [3:0] red,

## output reg [3:0] green,

## output reg [3:0] blue,

## input wire s3,

## input wire s2,

## input wire s1,

## input wire s0

## );

## 

## parameter hpixels = 800;

## parameter vlines = 521;

## parameter hpulse = 96;

## parameter vpulse = 2;

## parameter hbp = 144;

## parameter hfp = 784;

## parameter vbp = 31;

## parameter vfp = 511;

## reg [9:0] hc;

## reg [9:0] vc;

## reg [1:0] pxclk;

## wire inH = (hc < 640);

## wire inV = (vc < 480);

## wire inDisplay = inH && inV;

## always @ (posedge clk) pxclk = pxclk + 1;

## wire pclk;

## assign pclk = pxclk[1];

## 

## always @ (posedge pclk or posedge clr)

## begin

## if (clr == 1)

## begin

## hc <= 0;

## vc <= 0;

## end

## else

## begin

## if (hc < hpixels - 1)

## hc <= hc + 1;

## else

## begin

## hc <= 0;

## if (vc < vlines - 1)

## vc <= vc + 1;

## else

## vc <= 0;

## end

## end

## end

## assign hsync = (hc < hpulse) ? 0:1;

## assign vsync = (vc < vpulse) ? 0:1;

## wire [2:0] color;

## assign color = {s2, s1, s0};

## reg[11:0] c1 = 12'h000, c2=12'h00F, c3=12'h0F0, c4=12'h0FF, c5=12'hF00, c6=12'hF0F, c7=12'hFF0, c8=12'hFFF;

## always @ (vc or hc or inDisplay)

## begin

## if (vc >= vbp & vc < vfp & hc >= hbp & hc < hfp)

## begin

## case(color)

## 3'b000 : begin

## draw(0,0,639,479,c1);

## end

## 3'b001 : begin

## draw(0,0,639,479,c2);

## end

## 3'b010 : begin

## draw(0,0,639,479,c3);

## end

## 3'b011 : begin

## draw(0,0,639,479,c4);

## end

## 3'b100 : begin

## draw(0,0,639,479,c5);

## end

## 3'b101 : begin

## draw(0,0,639,479,c6);

## end

## 3'b110 : begin

## draw(0,0,639,479,c7);

## end

## 3'b111 : begin

## draw(0,0,639,479,c8);

## end

## default : draw(0,0,0,0,12'h000);

## endcase

## if( s3 == 1'b1 ) begin

## char(18, 220, 170); //s

## end

## if (!inDisplay)

## begin

## red = 0;

## green = 0;

## blue = 0;

## end

## end

## end

## function draw;

## input [9:0] xStart;

## input [9:0] yStart;

## input [9:0] xEnd;

## input [9:0] yEnd;

## input [11:0] color;

## begin

## if (vc >= (vbp + yStart) && vc < (vbp + yEnd) && hc >= (hbp + xStart) && hc < (hbp + xEnd))

## begin

## red = color[3:0];

## green = color[7:4];

## blue = color[11:8];

## end

## if (!inDisplay)

## begin

## red = 0;

## green = 0;

## blue = 0;

## end

## end

## endfunction

## function char;

## input [5:0] charVal;

## input [9:0] x, y;

## reg [11:0] white = 12'hFFF;

## begin

## case(charVal)

## 6'b000000: // A

## begin

## draw(x + 2, y, x + 5, y + 1, white);

## draw(x + 1, y + 1, x + 6, y + 2, white);

## draw(x, y + 2, x + 2, y + 9, white);

## draw(x + 2, y + 4, x + 5, y + 6, white);

## draw(x + 5, y + 2, x + 7, y + 9, white);

## end

## 6'b000001: // B

## begin

## draw(x, y, x + 5, y + 2, white);

## draw(x, y + 2, x + 2, y + 9, white);

## draw(x + 5, y + 2, x + 7, y + 4, white);

## draw(x + 2, y + 4, x + 5, y + 5, white);

## draw(x + 5, y + 5, x + 7, y + 8, white);

## draw(x + 2, y + 8, x + 5, y + 9, white);

## end

## 6'b000010: // C

## begin

## draw(x + 2, y, x + 6, y + 1, white);

## draw(x + 1, y + 1, x + 2, y + 2, white);

## draw(x + 6, y + 1, x + 7, y + 2, white);

## draw(x, y + 2, x + 2, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## draw(x + 6, y + 7, x + 7, y + 8, white);

## end

## 6'b000011: // D

## begin

## draw(x, y, x + 5, y + 1, white);

## draw(x, y + 1, x + 2, y + 9, white);

## draw(x + 5, y + 1, x + 6, y + 2, white);

## draw(x + 6, y + 2, x + 7, y + 7, white);

## draw(x + 5, y + 7, x + 6, y + 8, white);

## draw(x + 2, y + 8, x + 5, y + 9, white);

## end

## 6'b000100: // E

## begin

## draw(x, y, x + 7, y + 1, white);

## draw(x, y + 1, x + 2, y + 9, white);

## draw(x + 2, y + 4, x + 5, y + 5, white);

## draw(x + 2, y + 8, x + 7, y + 9, white);

## end

## 6'b000101: // F

## begin

## draw(x, y, x + 7, y + 1, white);

## draw(x, y + 1, x + 2, y + 9, white);

## draw(x + 2, y + 3, x + 5, y + 4, white);

## end

## 6'b000110: // G

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 2, y + 8, white);

## draw(x + 6, y + 1, x + 7, y + 2, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## draw(x + 5, y + 5, x + 7, y + 8, white);

## draw(x + 4, y + 5, x + 5, y + 6, white);

## end

## 6'b000111: // H

## begin

## draw(x, y, x + 2, y + 9, white);

## draw(x + 2, y + 4, x + 5, y + 5, white);

## draw(x + 5, y, x + 7, y + 9, white);

## end

## 6'b001000: // I

## begin

## draw(x, y, x + 2, y + 9, white);

## end

## 6'b001001: // J

## begin

## draw(x + 5, y, x + 7, y + 8, white);

## draw(x, y + 6, x + 1, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b001010: // K

## begin

## draw(x, y, x + 2, y + 9, white);

## draw(x + 2, y + 4, x + 5, y + 5, white);

## draw(x + 5, y + 3, x + 6, y + 4, white);

## draw(x + 5, y + 5, x + 6, y + 6, white);

## draw(x + 6, y, x + 7, y + 3, white);

## draw(x + 6, y + 6, x + 7, y + 9, white);

## end

## 6'b001011: // L

## begin

## draw(x, y, x + 2, y + 9, white);

## draw(x + 2, y + 7, x + 7, y + 9, white);

## end

## 6'b001100: // M

## begin

## draw(x + 1, y, x + 3, y + 1, white);

## draw(x + 4, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 7, y + 2, white);

## draw(x, y + 2, x + 2, y + 9, white);

## draw(x + 3, y + 2, x + 4, y + 9, white);

## draw(x + 5, y + 2, x + 7, y + 9, white);

## end

## 6'b001101: // N

## begin

## draw(x, y, x + 2, y + 9, white);

## draw(x + 2, y + 1, x + 3, y + 2, white);

## draw(x + 3, y, x + 6, y + 1, white);

## draw(x + 6, y + 1, x + 7, y + 9, white);

## end

## 6'b001110: // O

## begin

## draw(x, y + 1, x + 2, y + 8, white);

## draw(x + 5, y + 1, x + 7, y + 8, white);

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b001111: // P

## begin

## draw(x, y, x + 5, y + 1, white);

## draw(x, y + 1, x + 2, y + 9, white);

## draw(x + 5, y + 1, x + 6, y + 2, white);

## draw(x + 6, y + 2, x + 7, y + 4, white);

## draw(x + 5, y + 4, x + 6, y + 5, white);

## draw(x + 2, y + 5, x + 5, y + 6, white);

## end

## 6'b010000: // Q

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 2, y + 8, white);

## draw(x + 5, y + 1, x + 7, y + 7, white);

## draw(x + 1, y + 8, x + 7, y + 9, white);

## draw(x + 3, y + 6, x + 4, y + 7, white);

## draw(x + 4, y + 7, x + 6, y + 8, white);

## draw(x + 1, y + 8, x + 7, y + 9, white);

## end

## 6'b010001: // R

## begin

## draw(x, y, x + 5, y + 1, white);

## draw(x, y + 1, x + 2, y + 9, white);

## draw(x + 5, y + 1, x + 6, y + 2, white);

## draw(x + 6, y + 2, x + 7, y + 4, white);

## draw(x + 5, y + 4, x + 6, y + 5, white);

## draw(x + 2, y + 5, x + 6, y + 6, white);

## draw(x + 6, y + 6, x + 7, y + 9, white);

## end

## 6'b010010: // S

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 1, y + 4, white);

## draw(x + 6, y + 1, x + 7, y + 3, white);

## draw(x + 1, y + 4, x + 6, y + 5, white);

## draw(x + 6, y + 5, x + 7, y + 8, white);

## draw(x, y + 6, x + 1, y + 8, white);

## draw(x + 2, y + 8, x + 7, y + 9, white);

## end

## 6'b010011: // T

## begin

## draw(x, y, x + 7, y + 2, white);

## draw(x + 3, y + 2, x + 5, y + 9, white);

## end

## 6'b010100: // U

## begin

## draw(x, y, x + 2, y + 8, white);

## draw(x + 5, y, x + 7, y + 8, white);

## draw(x + 1, y + 7, x + 6, y + 9, white);

## end

## 6'b010101: // V

## begin

## draw(x, y, x + 1, y + 5, white);

## draw(x + 6, y, x + 7, y + 5, white);

## draw(x + 1, y + 4, x + 2, y + 7, white);

## draw(x + 5, y + 4, x + 6, y + 7, white);

## draw(x + 3, y + 6, x + 4, y + 8, white);

## draw(x + 5, y + 6, x + 6, y + 8, white);

## draw(x + 4, y + 8, x + 5, y + 9, white);

## end

## 6'b010110: // W

## begin

## draw(x, y, x + 1, y + 8, white);

## draw(x + 6, y, x + 7, y + 8, white);

## draw(x + 3, y + 3, x + 4, y + 8, white);

## draw(x, y + 8, x + 7, y + 9, white);

## end

## 6'b010111: // X

## begin

## draw(x, y, x + 1, y + 3, white);

## draw(x + 6, y, x + 7, y + 3, white);

## draw(x + 1, y + 2, x + 2, y + 4, white);

## draw(x + 5, y + 2, x + 6, y + 4, white);

## draw(x + 2, y + 4, x + 5, y + 5, white);

## draw(x + 1, y + 5, x + 2, y + 7, white);

## draw(x + 5, y + 5, x + 6, y + 7, white);

## draw(x, y + 6, x + 1, y + 9, white);

## draw(x + 6, y + 6, x + 7, y + 9, white);

## end

## 6'b011000: // Y

## begin

## draw(x, y, x + 2, y + 4, white);

## draw(x + 5, y, x + 7, y + 8, white);

## draw(x + 1, y + 4, x + 5, y + 5, white);

## draw(x, y + 7, x + 2, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b011001: // Z

## begin

## draw(x, y, x + 7, y + 2, white);

## draw(x + 5, y + 2, x + 7, y + 3, white);

## draw(x + 4, y + 3, x + 5, y + 4, white);

## draw(x + 3, y + 4, x + 4, y + 5, white);

## draw(x + 2, y + 5, x + 3, y + 6, white);

## draw(x, y + 6, x + 2, y + 7, white);

## draw(x + 1, y + 7, x + 8, y + 9, white);

## end

## 6'b011010: // 1

## begin

## draw(x + 3, y, x + 5, y + 9, white);

## draw(x + 2, y + 1, x + 3, y + 3, white);

## draw(x + 1, y + 2, x + 2, y + 3, white);

## draw(x + 1, y + 7, x + 6, y + 9, white);

## end

## 6'b011011: // 2

## begin

## draw(x, y + 1, x + 2, y + 3, white);

## draw(x + 1, y, x + 3, y + 2, white);

## draw(x + 3, y, x + 6, y + 1, white);

## draw(x + 5, y + 1, x + 7, y + 4, white);

## draw(x + 1, y + 4, x + 6, y + 5, white);

## draw(x, y + 5, x + 2, y + 8, white);

## draw(x + 2, y + 8, x + 8, y + 9, white);

## end

## 6'b011100: // 3

## begin

## draw(x, y + 1, x + 1, y + 2, white);

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x + 6, y + 1, x + 7, y + 4, white);

## draw(x + 2, y + 4, x + 6, y + 5, white);

## draw(x + 6, y + 5, x + 7, y + 8, white);

## draw(x, y + 7, x + 1, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b011101: // 4

## begin

## draw(x + 4, y, x + 7, y + 2, white);

## draw(x + 3, y + 1, x + 4, y + 2, white);

## draw(x + 2, y + 2, x + 3, y + 3, white);

## draw(x + 1, y + 3, x + 2, y + 4, white);

## draw(x, y + 4, x + 5, y + 6, white);

## draw(x + 4, y + 2, x + 7, y + 9, white);

## end

## 6'b011110: // 5

## begin

## draw(x, y, x + 7, y + 2, white);

## draw(x, y + 2, x + 2, y + 5, white);

## draw(x + 2, y + 3, x + 7, y + 5, white);

## draw(x + 5, y + 5, x + 7, y + 9, white);

## draw(x, y + 7, x + 7, y + 9, white);

## end

## 6'b011111: // 6

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 2, y + 8, white);

## draw(x + 5, y + 1, x + 7, y + 2, white);

## draw(x + 2, y + 4, x + 6, y + 5, white);

## draw(x + 5, y + 5, x + 7, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b100000: // 7

## begin

## draw(x, y, x + 7, y + 2, white);

## draw(x, y + 2, x + 2, y + 3, white);

## draw(x + 5, y + 2, x + 7, y + 9, white);

## end

## 6'b100001: // 8

## begin

## draw(x + 1, y, x + 6, y + 2, white);

## draw(x, y + 1, x + 2, y + 4, white);

## draw(x + 5, y + 1, x + 7, y + 4, white);

## draw(x + 1, y + 4, x + 6, y + 5, white);

## draw(x, y + 5, x + 2, y + 8, white);

## draw(x + 6, y + 5, x + 8, y + 8, white);

## draw(x + 2, y + 7, x + 7, y + 9, white);

## end

## 6'b100010: // 9

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 2, y + 4, white);

## draw(x + 1, y + 4, x + 5, y + 5, white);

## draw(x + 5, y + 1, x + 7, y + 8, white);

## draw(x, y + 7, x + 1, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## 6'b100011: // 0

## begin

## draw(x + 1, y, x + 6, y + 1, white);

## draw(x, y + 1, x + 7, y + 2, white);

## draw(x, y + 2, x + 2, y + 7, white);

## draw(x + 5, y + 2, x + 7, y + 7, white);

## draw(x, y + 7, x + 7, y + 8, white);

## draw(x + 1, y + 8, x + 6, y + 9, white);

## end

## default:

## begin

## red = 0;

## green = 0;

## blue = 0;

## end

## endcase

## end

## endfunction

## endmodule

## User Constraints:

## # Pin assignment

## set\_property -dict { PACKAGE\_PIN A4 IOSTANDARD LVCMOS33 } [get\_ports { red[3] }]; # VGA\_R3

## set\_property -dict { PACKAGE\_PIN C5 IOSTANDARD LVCMOS33 } [get\_ports { red[2] }]; # VGA\_R2

## set\_property -dict { PACKAGE\_PIN B4 IOSTANDARD LVCMOS33 } [get\_ports { red[1] }]; # VGA\_R1

## set\_property -dict { PACKAGE\_PIN A3 IOSTANDARD LVCMOS33 } [get\_ports { red[0] }]; # VGA\_R0

## set\_property -dict { PACKAGE\_PIN A6 IOSTANDARD LVCMOS33 } [get\_ports { green[3] }]; # VGA\_G3

## set\_property -dict { PACKAGE\_PIN B6 IOSTANDARD LVCMOS33 } [get\_ports { green[2] }]; # VGA\_G2

## set\_property -dict { PACKAGE\_PIN A5 IOSTANDARD LVCMOS33 } [get\_ports { green[1] }]; # VGA\_G1

## set\_property -dict { PACKAGE\_PIN C6 IOSTANDARD LVCMOS33 } [get\_ports { green[0] }]; # VGA\_G0

## set\_property -dict { PACKAGE\_PIN D8 IOSTANDARD LVCMOS33 } [get\_ports { blue[3] }]; # VGA\_B3

## set\_property -dict { PACKAGE\_PIN D7 IOSTANDARD LVCMOS33 } [get\_ports { blue[2] }]; # VGA\_B2

## set\_property -dict { PACKAGE\_PIN C7 IOSTANDARD LVCMOS33 } [get\_ports { blue[1] }]; # VGA\_B1

## set\_property -dict { PACKAGE\_PIN B7 IOSTANDARD LVCMOS33 } [get\_ports { blue[0] }]; # VGA\_B0

## set\_property -dict { PACKAGE\_PIN B11 IOSTANDARD LVCMOS33 } [get\_ports { hsync }]; # VGA\_HS

## set\_property -dict { PACKAGE\_PIN B12 IOSTANDARD LVCMOS33 } [get\_ports { vsync }]; # VGA\_VS

## set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; # CLK100MHZ

## # Clock definition

## create\_clock -name sys\_clk -period 10.00 [get\_ports {clk}]; # 100 MHz

## set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { clr }];

## # Configuration Bank Voltage Select

## set\_property CFGBVS VCCO [current\_design]

## set\_property CONFIG\_VOLTAGE 3.3 [current\_design]

## #switches

## set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { s0 }];

## set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { s1 }];

## set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { s2 }];

## set\_property -dict { PACKAGE\_PIN R17 IOSTANDARD LVCMOS33 } [get\_ports { s3 }];

## Results Discussion:

## We got the results as expected. The FPGA outputs a VGA signal as mostly a solid color. The VGA signal changes colors, with the pictures showing red and 15 blue respectively on the display. With the switches the FPGA properly shows all 8 colors (Red, Green, Blue, Yellow, Cyan, Magenta, Black, and White) but didn’t take the picture of all of the colors. The picture also shows the character S as the lab specified, the character can be turned on and off with the switch.

Figure 4: The figure below are the colors that were displayed on the screen and the first letter of my name.

A screen shot of a red door

Description automatically generatedA picture containing wall, monitor, electronics, indoor

Description automatically generatedA picture containing wall, indoor, electronics, sitting

Description automatically generatedA close up of a door

Description automatically generatedA close up of a screen door

Description automatically generatedA screen shot of a television

Description automatically generatedA screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

Conclusion:

In this lab, we learn how to output a VGA Signal, through outputting colors and outputting the first letter of our name. how to use the VGA output and how improve my skills on FPGA programming in general. Part 1 was VGA Display of different colors, where we manipulated the RGB signals to change the signals to different colors. Part 2 was character display on FPGA, where we specified specific characters to be a specific color, so we can draw a character on the display. After this lab I learned learn how to use the VGA output and improved my skills on FPGA programming in general. Overall, the lab was straightforward and went well.

## 