**ECEN 323 – Winter 2020**

Lab 3: VGA Object Display

Johnson, Ryan

Section 01

Preliminary

Why doesn't the Seven Segment Controller just display all eight digits at the same time?

The cathode signals are shared so if all digits are on at once, they will all display the same number. One must cycle between each digit to show the individual digits correctly one at a time. This happens so quickly that it looks like they are all being displayed at once.

How long (in microseconds) is each digit illuminated for?

0.32 microseconds

What is the purpose of the 'anode' signals of the seven-segment controller?

There is an anode signal for each of the eight digits to tell it whether to display the segment cathode signals.

What is the purpose of the 'segment' cathode signals of the seven-segment controller?

There is a segment cathode signal for each of the segments on the digit lettered a-g. The cathode signals are shared among all eight digits.

Exercise #1

Submit your Verilog code segment for this example

`timescale 1ns / 1ps

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Module: vga\_test

\*

\* Author: Ryan Johnson

\* Class: ECEN 323, Section 01, Winter 2020

\* Date: 24 January 2020

\*

\* Description: module to display four colored squares.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

module vga\_test(

input wire logic clk,

input wire logic btnc,

output logic hs,

output logic vs,

output logic [3:0] vgaRed,

output logic [3:0] vgaGreen,

output logic [3:0] vgaBlue

);

logic hsreg;

logic vsreg;

logic [9:0] pixel\_x;

logic [9:0] pixel\_y;

logic last\_column;

logic last\_row;

logic blank;

//instantiating our vga\_timing module to get the various values through

vga\_timing v1(.clk(clk), .rst(btnc), .hs(hsreg), .vs(vsreg), .pixel\_x(pixel\_x), .pixel\_y(pixel\_y), .last\_column(last\_column), .last\_row(last\_row), .blank(blank) );

//register for hs to avoid glitches in timing signals

always\_ff@(posedge clk)

hs <= hsreg;

//register for vs to avoid glitches in timing signals

always\_ff@(posedge clk)

vs <= vsreg;

logic red\_square\_on;

logic [11:0] red\_square\_rgb;

logic green\_square\_on;

logic [11:0] green\_square\_rgb;

logic yellow\_square\_on;

logic [11:0] yellow\_square\_rgb;

logic magenta\_square\_on;

logic [11:0] magenta\_square\_rgb;

logic [11:0] background\_rgb;

logic [11:0] rgb\_out;

assign red\_square\_on = ((pixel\_x >= 180 && pixel\_x < 280) &&

(pixel\_y >= 100 && pixel\_y < 200));

assign red\_square\_rgb = {4'b1111,4'b0000,4'b0000};

assign green\_square\_on = ((pixel\_x >= 400 && pixel\_x < 500) &&

(pixel\_y >= 100 && pixel\_y < 200));

assign green\_square\_rgb = {4'b0000,4'b1111,4'b0000};

assign yellow\_square\_on = ((pixel\_x >= 180 && pixel\_x < 280) &&

(pixel\_y >= 300 && pixel\_y < 400));

assign yellow\_square\_rgb = {4'b1111,4'b1111,4'b0000};

assign magenta\_square\_on = ((pixel\_x >= 400 && pixel\_x < 500) &&

(pixel\_y >= 300 && pixel\_y < 400));

assign magenta\_square\_rgb = {4'b1111,4'b0000,4'b1111};

assign background\_rgb = {4'b1111,4'b1111,4'b1111};

assign rgb\_out = red\_square\_on ? red\_square\_rgb :

green\_square\_on ? green\_square\_rgb :

yellow\_square\_on ? yellow\_square\_rgb :

magenta\_square\_on ? magenta\_square\_rgb :

background\_rgb;

always\_ff@(posedge clk)

begin

vgaRed <= rgb\_out[11:8];

vgaGreen <= rgb\_out[7:4];

vgaBlue <= rgb\_out[3:0];

end

endmodule

Exercise #2

Submit your top-level Verilog code

`timescale 1ns / 1ps

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Module: vga\_object

\*

\* Author: Ryan Johnson

\* Class: ECEN 323, Section 01, Winter 2020

\* Date: 24 January 2020

\*

\* Description: top module for displaying on vga according to buttons pressed.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

module vga\_object(

input wire clk,

input wire btnc,

input wire btnl,

input wire btnr,

input wire btnd,

input wire btnu,

input wire [11:0] sw,

output logic hs,

output logic vs,

output logic [3:0] vgaRed,

output logic [3:0] vgaGreen,

output logic [3:0] vgaBlue,

output logic [7:0] seg,

output logic [7:0] anode

);

logic hsreg;

logic vsreg;

logic [9:0] pixel\_x;

logic [9:0] pixel\_y;

logic last\_column;

logic last\_row;

logic blank;

//instantiating our vga\_timing module to get the various values through

vga\_timing v1(.clk(clk), .rst(btnc), .hs(hsreg), .vs(vsreg), .pixel\_x(pixel\_x), .pixel\_y(pixel\_y), .last\_column(last\_column), .last\_row(last\_row), .blank(blank) );

//register for hs and vs to avoid glitches in timing signals

always\_ff@(posedge clk)

begin

hs <= hsreg;

vs <= vsreg;

end

localparam CLOCKS\_PER\_PIXEL\_CLK = 4;

logic [1:0] pixel\_cnt;

logic [15:0] frame\_cnt;

logic frame\_en;

always\_ff@(posedge clk)

begin

if (btnc)

pixel\_cnt <= 0;

else

pixel\_cnt <= pixel\_cnt + 1;

end

assign frame\_en = (pixel\_cnt == CLOCKS\_PER\_PIXEL\_CLK - 1) ? 1'b1 : 1'b0;

always\_ff@(posedge clk)

begin

if(btnc)

frame\_cnt <= 0;

else if(frame\_en && last\_row && last\_column)

frame\_cnt <= frame\_cnt + 1;

end

logic[31:0] dataIn = {16'h0000,frame\_cnt};

SevenSegmentControl ss1(.clk(clk),.reset(btnc),.dataIn(dataIn),.digitDisplay(8'h0F),.digitPoint(8'h00),.anode(anode),.segment(seg));

logic red\_square\_on;

logic [11:0] red\_square\_rgb;

logic green\_square\_on;

logic [11:0] green\_square\_rgb;

logic yellow\_square\_on;

logic [11:0] yellow\_square\_rgb;

logic magenta\_square\_on;

logic [11:0] magenta\_square\_rgb;

logic [11:0] background\_rgb;

logic [11:0] blank\_rgb;

logic [11:0] nb\_rgb\_out;

logic [11:0] lb\_rgb\_out;

logic [11:0] rb\_rgb\_out;

logic [11:0] ub\_rgb\_out;

logic [11:0] db\_rgb\_out;

logic [11:0] rgb\_out;

localparam MAX = 4'b1111;

localparam MIN = 4'b0000;

assign blank\_rgb = {MIN,MIN,MIN};

assign ub\_rgb\_out = {MIN,MIN,MIN};

assign db\_rgb\_out = {MAX,MAX,MAX};

assign rb\_rgb\_out = sw;

assign red\_square\_on = ((pixel\_x >= 180 && pixel\_x < 280) &&

(pixel\_y >= 100 && pixel\_y < 200));

assign red\_square\_rgb = {MAX,MIN,MIN};

assign green\_square\_on = ((pixel\_x >= 400 && pixel\_x < 500) &&

(pixel\_y >= 100 && pixel\_y < 200));

assign green\_square\_rgb = {MIN,MAX,MIN};

assign yellow\_square\_on = ((pixel\_x >= 180 && pixel\_x < 280) &&

(pixel\_y >= 300 && pixel\_y < 400));

assign yellow\_square\_rgb = {MAX,MAX,MIN};

assign magenta\_square\_on = ((pixel\_x >= 400 && pixel\_x < 500) &&

(pixel\_y >= 300 && pixel\_y < 400));

assign magenta\_square\_rgb = {MAX,MIN,MAX};

assign background\_rgb = {MAX,MAX,MAX};

assign nb\_rgb\_out = red\_square\_on ? red\_square\_rgb :

green\_square\_on ? green\_square\_rgb :

yellow\_square\_on ? yellow\_square\_rgb :

magenta\_square\_on ? magenta\_square\_rgb :

background\_rgb;

localparam BLACK = 79;

localparam BLUE = 159;

localparam GREEN = 239;

localparam CYAN = 319;

localparam RED = 399;

localparam MAGENTA = 479;

localparam YELLOW = 559;

logic [3:0] red\_color;

logic [3:0] green\_color;

logic [3:0] blue\_color;

assign red\_color = (pixel\_x <= CYAN) ? MIN : MAX;

assign green\_color = (pixel\_x <= BLUE) ? MIN :

(pixel\_x <= CYAN) ? MAX :

(pixel\_x <= MAGENTA)? MIN : MAX;

assign blue\_color = (pixel\_x <= BLACK) ? MIN :

(pixel\_x <= BLUE) ? MAX :

(pixel\_x <= GREEN) ? MIN :

(pixel\_x <= CYAN) ? MAX :

(pixel\_x <= RED) ? MIN :

(pixel\_x <= MAGENTA)? MAX :

(pixel\_x <= YELLOW) ? MIN : MAX;

assign lb\_rgb\_out = {red\_color, green\_color, blue\_color};

assign rgb\_out = blank ? blank\_rgb :

btnl ? lb\_rgb\_out :

btnr ? rb\_rgb\_out :

btnu ? ub\_rgb\_out :

btnd ? db\_rgb\_out :

nb\_rgb\_out;

always\_ff@(posedge clk)

begin

vgaRed <= rgb\_out[11:8];

vgaGreen <= rgb\_out[7:4];

vgaBlue <= rgb\_out[3:0];

end

endmodule

Submit your .tcl script

restart

add\_force clk {0 0} {1 5ns} -repeat\_every 10ns

add\_force btnc 1

run 300 ns

add\_force btnc 0

add\_force sw 0000101001011010

add\_force btnl 0

add\_force btnu 0

add\_force btnr 0

add\_force btnd 1

run 18000000 ns

add\_force btnd 0

add\_force btnr 1

run 18000000 ns

add\_force btnr 0

add\_force btnc 1

run 300 ns

add\_force btnc 0

run 18000000 ns

Exercise #3

Summarize any synthesis warnings you received. Indicate “none” if you had no warnings.

none

Summarize the resource utilization of your design by filling in the table below.

| **Resource** | **Utilization** |
| --- | --- |
| LUT | 73 |
| FF | 71 |
| BRAM | 0 |
| IO | 48 |
| BUFG | 1 |

How many hours did you work on the lab?

3 Hours

Please provide any suggestions for improving this lab in the future:

None. Everything made sense and was laid out in a way that was easy to grasp and understand the importance of each step.