



Lab 11 – VGA Controller

|  |  |  |
| --- | --- | --- |
| Name: Ali Muhammad Asad, Shayan Shoaib Patel | ID: aa07190, sp07101 | Section: T6 |

**Objectives**

The description of each section is given below:

|  |  |
| --- | --- |
| Section |  |
| a) Introduction | **10** |
| In this section, you will be introduced to VGA display and the required modules to design VGA controller. |
| b) Working of VGA controller  (Synchronization Circuit) | **50** |
| In this section, you will construct VGA synchronization circuit to generate *h\_sync*  and *v\_sync* signals |
| (Pixel Generation Circuit) | **30** |
| In this task, you will construct a pixel generation circuit. This circuit will determine the pixel to show the desired image in VGA Monitor |
| (VGA Controller Complete Circuit) | **30** |
| In this part, you will combine all the defined Verilog modules together to get the system working and test its functionality. |
| Exercise  You will modify pixel generation module to display different images on VGA monitor using Basys-3 Board | **30** |

# Introduction

VGA (Video Graphics Array) is a video display standard introduced in the late 1980s in IBM PCs and allowed for a display resolution of 640x480 pixels it is widely supported by PC graphics hardware and monitors.

You have already developed horizontal and vertical counter (*h\_counter*, *v\_counter*) and clock divider (*clock\_div*) modules in previous. In this lab, you are going to develop remaining modules of VGA Controller. A video controller generates the synchronization signals and outputs data pixels. A complete block diagram of a VGA controller is shown in [Figure 11. 7](#_bookmark5). It contains an additional synchronization circuit, labeled *vga\_sync*, and a pixel generation circuit labelled *pixel\_gen*.

# Working of VGA controller

A VGA monitor operates using an electron beam that scans the screen row by row, starting at the upper left corner and ending at the lower-right corner as shown in [Figure 11. 1.](#_bookmark0) This beam moves using two synchronization signals, called *h\_sync* (horizontal synchronization), and *v\_sync* (vertical synchronization). The *h\_sync* signal tells the beam when to move to the next row. The *v\_sync* signal tells the beam when to move to back to the top of the screen. To display a picture on the screen, we simply generate these synchronization signals and provide the pixel color to display on the screen.

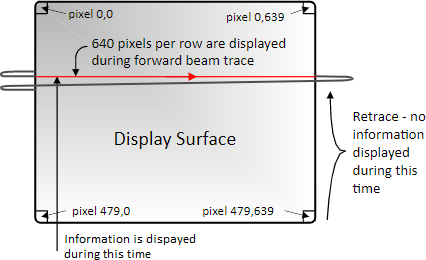


Figure 11. 1

## Synchronization Circuit (vga\_sync)

The *vga\_sync* circuit generates timing and synchronization signals. It takes *h\_count* and

*v\_count* as input and generate five outputs as shown in [Figure 11. 2.](#_bookmark1)

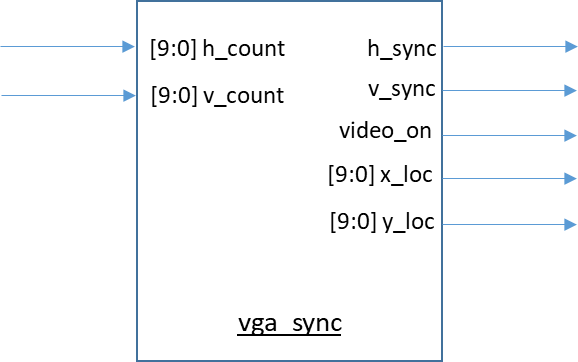


Figure 11. 2

* The *h\_sync* and *v\_sync* signals are connected to the VGA port to control the horizontal and vertical scans of the monitor in transverse manner. (details will be discussed in next heading)
* the *video\_on* signal indicate whether the current targeted pixel is in the displayable region. It is asserted only when the *h\_count* smaller than 640 and *v\_count* is smaller than 480.
* The *x\_loc* (the location of pixel x) *y\_loc* (the location of pixel y) signals. The *x\_loc* and *y\_loc* specify the location of the current pixel and can be obtained from h\_count and v\_count.

The screen of a monitor includes a small black border, as shown at the top of [Figure 11. 3](#_bookmark2). The middle rectangle is the visible portion. Note that the coordinate of the vertical axis increases downward. The coordinates of the top-left and bottom-right comers are (0,0) and (639,479), respectively.

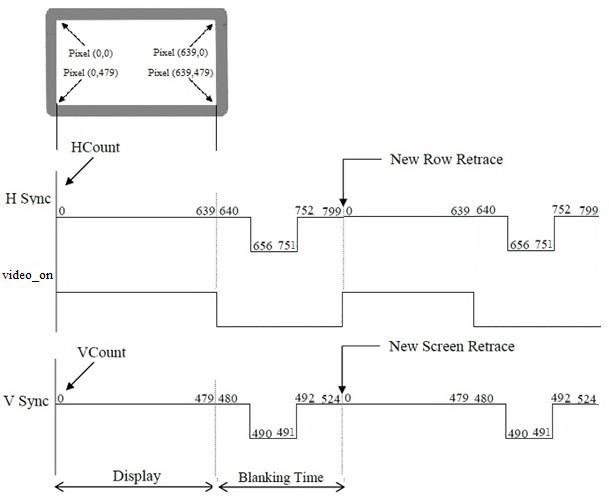


Figure 11. 3

*Horizontal synchronization (h\_sync)*

A detailed timing diagram of one horizontal scan is shown in [Figure 11. 3](#_bookmark2). A period of the

*h\_sync* signal contains 800 pixels and can be divided into two regions:

* *Display*: region where the pixels are actually displayed on the screen. The length of this region is 640 pixels.
* *Blanking:* region has 3 segments
  + *Retrace*: region in which the electron beams return to the left edge. The video signal should be disabled (i.e., black), and the length of this region is 96 pixels.
  + *Right border*: region that forms the right border of the display region. It is also known as the front porch (i.e., porch before retrace). The video signal should be disabled, and the length of this region is 16 pixels.
  + *Left border*: region that forms the left border of the display region. It is also known as the back porch (i.e., porch after retrace). The video signal should be disabled, and the length of this region is 48 pixels.

Note that the lengths of the right and left borders may vary for different brands of monitors. The *h\_sync* signal is obtained by *h\_counter* (mod-800 counter designed in previous lab). We intentionally start the counting from the beginning of the display region. This allows us to use the counter output as the horizontal (x-axis) coordinate. This output constitutes the *x\_loc* signal. The *h\_sync* signal goes low when the counter's output is between 656 and 751.

*Vertical synchronization (v\_sync)*

During the vertical scan, the electron beams move gradually from top to bottom and then return to the top. This corresponds to the time required to refresh the entire screen. The format of the *v\_sync* signal is similar to that of the *h\_sync* signal, as shown in [Figure 11. 3](#_bookmark2). The time unit of the movement is represented in terms of horizontal scan lines. A period of the *v\_sync* signal is 525 lines and can be divided into two regions:

* *Display*: region where the horizontal lines are actually displayed on the screen. The length of this region is 480 lines.
* *Blanking*: region has 3 segments
  + *Retrace*: region that the electron beams return to the top of the screen. The video signal should be disabled, and the length of this region is 2 lines.
  + *Bottom border*: region that forms the bottom border of the display region. It is also known as the front porch (i.e., porch before retrace). The video signal should be disabled, and the length of this region is 10 lines.
  + *Top border*: region that forms the top border of the display region. It is also known as the back porch (i.e., porch after retrace). The video signal should be disabled, and the length of this region is 33 lines.

As in the horizontal scan, the lengths of the top and bottom borders may vary for different brands of monitors.

The *v\_sync* signal can be obtained by *v\_counter* (mod-525 counter designed in previous lab). Again, we intentionally start counting from the beginning of the display region. This allows us to use the counter output as the vertical (y-axis) coordinate. This output constitutes the pixel-y signal. The *v\_sync* signal goes low when the line count is 490 and 491.

*Task A:*

Design a vga\_sync module as mentioned in [Figure 11. 2](#_bookmark1). Follow the timing diagram in [Figure](#_bookmark2)

[11. 3](#_bookmark2) to write logic for h\_sync, v\_sync video\_on, x\_loc and y\_loc outputs. A skeleton code for vga\_sync is given in [Figure 11. 4](#_bookmark3).

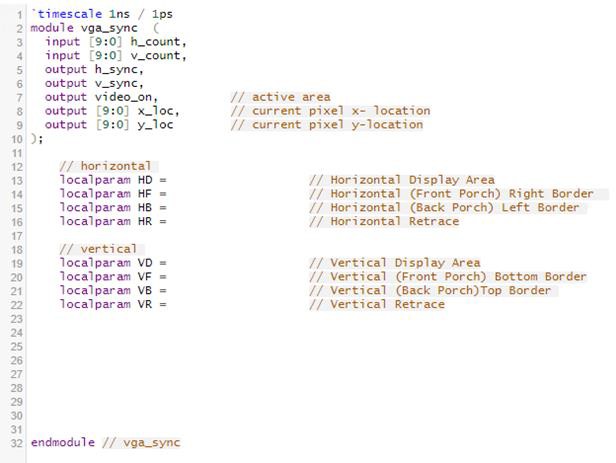


Figure 11. 4

|  |  |
| --- | --- |
|  | Verilog HDL local parameters are identical to parameters except that they cannot directly be modified by defparam statements or module instance parameter value assignments. Local parameters can be assigned constant expressions containing parameters, which can be modified with defparam statements or module instance  parameter value assignments. |

*Provide appropriately commented Verilog code here.*

`timescale 1ns / 1ps

module vga\_sync( input [9:0] h\_count,

input [9:0] v\_count, output h\_sync, output v\_sync,

output video\_on, //active area

output [9:0] x\_loc, //current pixel - x location output [9:0] y\_loc // current pixel - y location

);

//horizontal localparam HD = 640; localparam HF = 16; localparam HB = 48; localparam HR = 96;

//vertical

localparam VD = 480; localparam VF = 10; localparam VB = 33; localparam VR = 2;

assign v\_sync = (v\_count < (VD+VF)) | (v\_count >= (VD+VF+VR)); assign h\_sync = (h\_count < (HD+HF)) | (h\_count >= (HD+HF+HR)); assign video\_on = (h\_count < HD) & (v\_count<VD);

assign x\_loc = h\_count; assign y\_loc = v\_count;

endmodule

## Pixel Generation Circuit

The pixel generation circuit generates the three 4-bit color signals, which are collectively referred to as the RGB signal.

A color value is obtained according to the current coordinates of the pixel (the *x\_loc* and *y\_loc*

signals) and the external control and data signals.

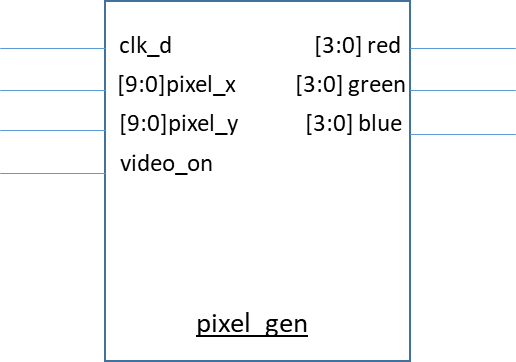


Figure 11. 5

*Task B*

Write down the pixel generator module shown in the [Figure 11. 6.](#_bookmark4) This module has one input clk\_d (of 25MHz), two 10-bit input pixel\_x and pixel\_y and one input video\_on. This module generates three 4-bit signals of Red, Green and Blue color.

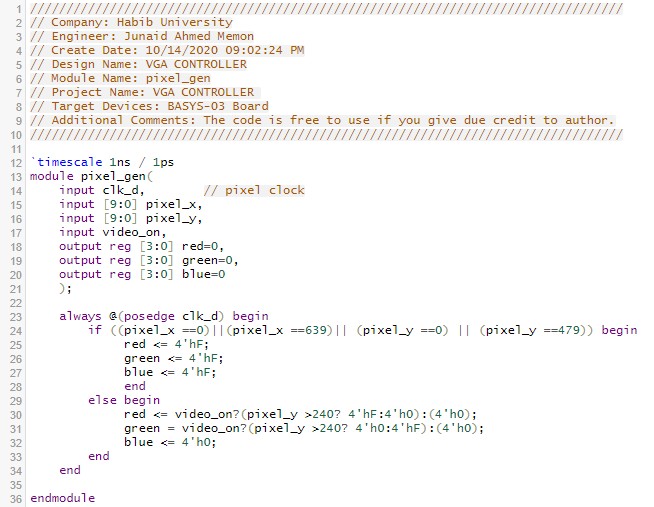
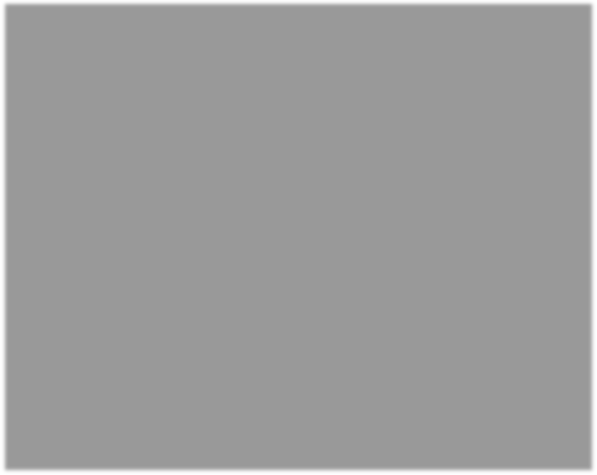


Figure 11. 6

This code generates horizontal stripes of green and red color on the screen (refer [Figure 11. 8](#_bookmark6)) with fine border of white color.

## VGA Controller Complete Circuit

This task we will connect all the modules designed in Lab 9 and 10 and test their functionality on VGA monitor.

*Task C*

1. Create a Top Level Module as in [Figure 11. 7.](#_bookmark5) This module will receive a 100Mhz Clock as input and generate 1-bit h\_sync and v\_sync signals and three 4-bit signals for Red Green and Blue color.

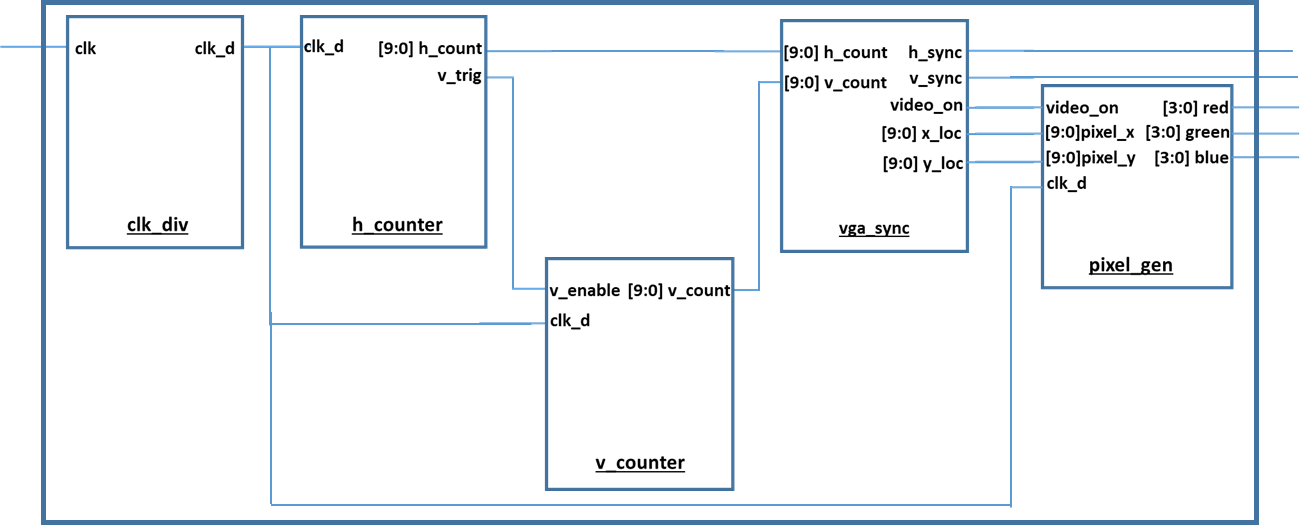


Figure 11. 7

*Provide appropriately commented Verilog code here.*

`timescale 1ns / 1ps

module TopLevelModule\_vga( input clk,

output h\_sync, output v\_sync, output [3:0] red,

output [3:0] green,

output [3:0] blue

);

wire clk\_d; wire trig\_v; wire video\_on;

wire [9:0] h\_count; wire [9:0] v\_count; wire [9:0] x\_loc; wire [9:0] y\_loc;

clk\_div clkd(.clk(clk), .clk\_d(clk\_d));

h\_counter hc(.clk(clk\_d), .h\_count(h\_count), .trig\_v(trig\_v)); v\_counter vc(.clk(clk\_d), .enable\_v(trig\_v), .v\_counter(v\_count));

vga\_sync vgas(.h\_count(h\_count), .v\_count(v\_count), .h\_sync(h\_sync), .v\_sync(v\_sync), .video\_on (video\_on), .x\_loc(x\_loc), .y\_loc(y\_loc));

pixel\_gen pg(.clk\_d(clk\_d), .pixel\_x(x\_loc), .pixel\_y(y\_loc), .video\_on(video\_on), .red(red), .blue (blue), .green(green));

endmodule

1. Implement1 the VGA controller on BASYS-03 board and show output on VGA monitor. The display output is shown in [Figure 11. 8](#_bookmark6). The VGA port connections are shown in [Figure 11. 9](#_bookmark7).

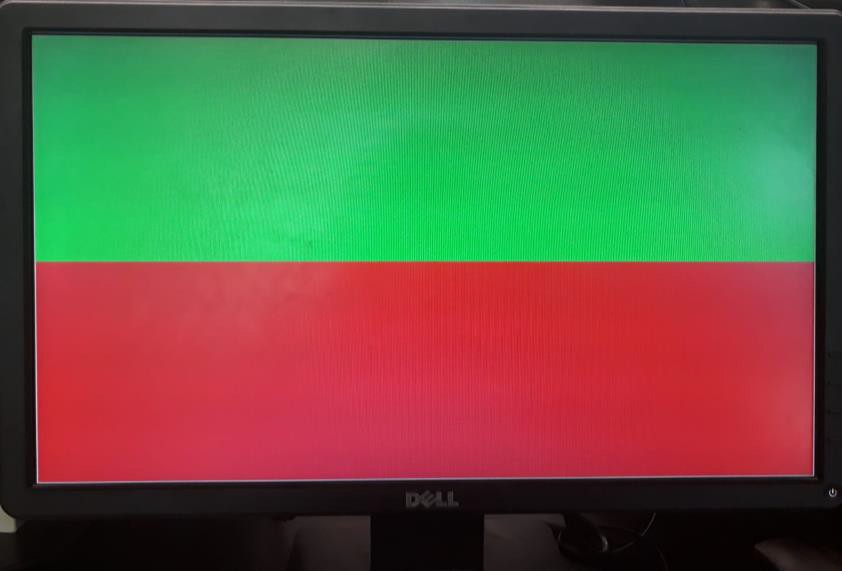


Figure 11. 8: VGA display

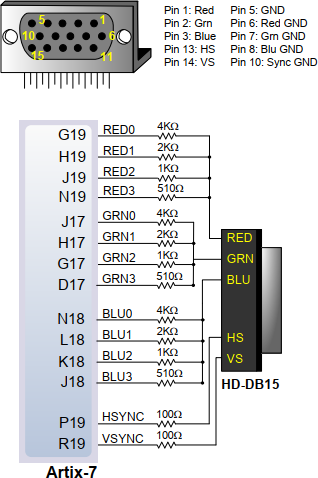


Figure 11. 9: VGA port connections

*Demonstrate your working VGA monitor to respective RA.*

1 Add your design files to Xilinx Vivado. Refering [Figure 11. 9,](#_bookmark7) create constraint file (.xdc file). For input clock signal, use W5 pin.



# Exercise

Modify *pixel\_gen* module to show an 8x8 checker board on VGA monitor as shown in [Figure](#_bookmark8)

[11. 10.](#_bookmark8) Each box on checked board should be a square (i.e. of equal width and height) you can fill the remaining area of screen with color of your choice. (Refer [Table 11. 1](#_bookmark9))

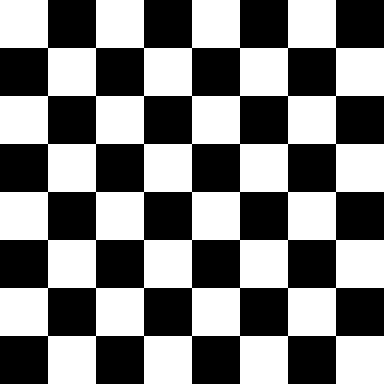


Figure 11. 10

Table 11. 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Three-bit VGA color combinations** | | | |
| **Red (R)** | **Green (G)** | **Blue (B)** | **Resulting color** |
| 0000 | 0000 | 0000 | black |
| 0000 | 0000 | 1111 | blue |
| 0000 | 1111 | 0000 | green |
| 0000 | 1111 | 1111 | cyan |
| 1111 | 0000 | 0000 | red |
| 1111 | 0000 | 1111 | magenta |
| 1111 | 1111 | 0000 | yellow |
| 1111 | 1111 | 1111 | white |



*Provide updated Verilog code for pixel generator here.*

`timescale 1ns / 1ps

module pixel\_gen(

input clk\_d, //pixel clock input [9:0] pixel\_x,

input [9:0] pixel\_y, input video\_on,

output reg [3:0] red = 0,

output reg [3:0] blue = 0,

output reg [3:0] green = 0

);

// CHECKERBOARD CODE \_ RECTANGULAR BOXES

always @(posedge clk\_d) begin if (

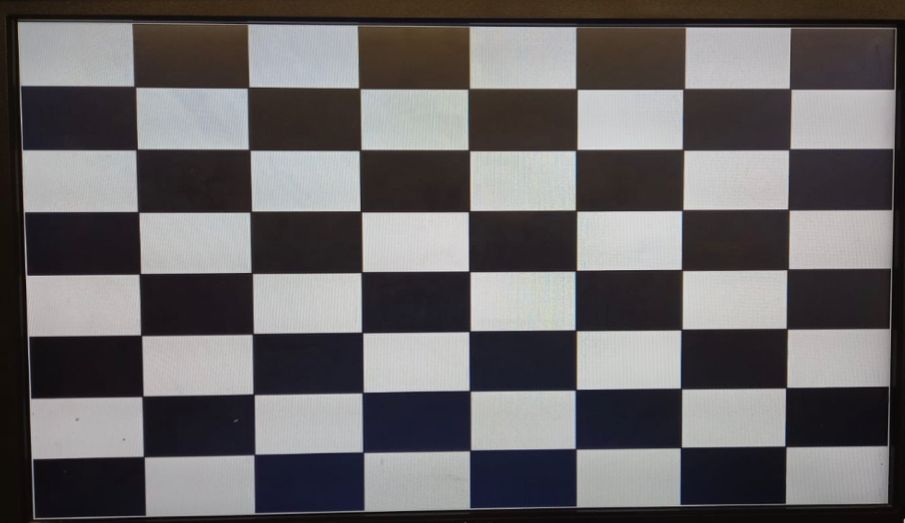
(pixel\_x >= 0 && pixel\_x <= 79 && pixel\_y >= 0 && pixel\_y <= 59) || (pixel\_x >= 160 && pixel\_x <= 239 && pixel\_y >= 0 && pixel\_y <= 59) || (pixel\_x >= 320 && pixel\_x <= 399 && pixel\_y >= 0 && pixel\_y <= 59) || (pixel\_x >= 480 && pixel\_x <= 559 && pixel\_y >= 0 && pixel\_y <= 59)||

(pixel\_x >= 80 && pixel\_x <= 159 && pixel\_y >= 60 && pixel\_y <= 119) || (pixel\_x >= 240 && pixel\_x <= 319 && pixel\_y >= 60 && pixel\_y <= 119) || (pixel\_x >= 400 && pixel\_x <= 479 && pixel\_y >= 60 && pixel\_y <= 119) || (pixel\_x >= 560 && pixel\_x <= 639 && pixel\_y >= 60 && pixel\_y <= 119) ||

(pixel\_x >= 0 && pixel\_x <= 79 && pixel\_y >= 120 && pixel\_y <= 179) || (pixel\_x >= 160 && pixel\_x <= 239 && pixel\_y >= 120 && pixel\_y <= 179) || (pixel\_x >= 320 && pixel\_x <= 399 && pixel\_y >= 120 && pixel\_y <= 179) || (pixel\_x >= 480 && pixel\_x <= 559 && pixel\_y >= 120 && pixel\_y <= 179)||

(pixel\_x >= 80 && pixel\_x <= 159 && pixel\_y >= 180 && pixel\_y <= 239) || (pixel\_x >= 240 && pixel\_x <= 319 && pixel\_y >= 180 && pixel\_y <= 239) || (pixel\_x >= 400 && pixel\_x <= 479 && pixel\_y >= 180 && pixel\_y <= 239) || (pixel\_x >= 560 && pixel\_x <= 639 && pixel\_y >= 180 && pixel\_y <= 239) ||

(pixel\_x >= 0 && pixel\_x <= 79 && pixel\_y >= 240 && pixel\_y <= 299) ||

*Show the VGA monitor output to your respective lab RA. Also attach display image.*

# Assessment Rubrics

**Marks Distribution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva** | **LR9**  **Report** |
| **In-lab** | **Task a** | 20 points | - | 10 points | 05 points |
| **Task b** | - | - |
| **Task c** | 10 points | 15 points |
| **Exercise** |  | 20 points | 20 points |
| **Total** | 100 points | | | | |

**Marks Obtained:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva** | **LR9**  **Report** |
| **In-lab** | **Task a** |  | - |  |  |
| **Task b** | - | - |
| **Task c** |  |  |
| **Exercise** |  |  |  |
| **Total** |  | | | | |

**Lab Evaluation Rubrics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Assessment Elements** | **Level 1: Unsatisfactory** | **Level 2: Developing** | **Level 3: Good** | **Level 4: Exemplary** |
| **LR2** | **Program/Co de/ Simulation Model/ Network Model** | Program/code/simulati on model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software. | Program/code/simula tion model/network model has some errors and does not produce completely accurate results.  Student has limited command on the basic tools of the software. | Program/code/simula tion model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine. | Program/code/simulati on /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software. |
| **LR5** | **Results & Plots** | Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner. | Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear. | All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing. | Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic. |
| **LR7** | **Viva** | Response shows a complete lack of understanding of the assigned / completed task | Response shows shallow understanding of the assigned task | Response shows substantial understanding of the assigned task | Response shows complete understanding of the completed task. The student is able to explain all the related concepts. |
| **LR9** | **Report** | No summary provided. The number/amount of tasks completed below the level of satisfaction and/or submitted late | Couldn’t provide good summary of in-lab tasks. Some major tasks were  completed but not explained well. Submission on time. Some major plots and figures provided | Good summary of In-lab tasks. All major tasks completed except few minor ones. The work is supported by some decent explanations, Submission on time, All necessary plots, and figures provided | Outstanding Summary of In-Lab tasks. All task completed and explained well, submitted on time, good presentation of plots and figure with proper label, titles and captions |