VGA DISPLAY USING FPGA

COMPUTER ORGANISATION & ARCHITECTURE

COMPUTER SCIENCE AND TECHNOLOGY

IIEST SHIBPUR

Course Responsible : SURAJEET GHOSH

Submitted By :

SOMNATH KARMAKAR , ROLL NO:13, Mtech 1st Year

SOUROJIT SEN , ROLL NO:15 , Mtech 1st Year

Batch : 2018-2020

ABSTRACT

This report summarizes the project 'Vga Display using Fpga' where we implemented 'print iiest' function using FPGA Board as the requirement of the 'Computer Organisation & Architecture Lab' Course . The report describes the implementation of the project with hardware description language (Verilog) using the Xilinx Platform Studio (XPS). It also includes the overall hardware and software overview and design details.

Table of Contents

- *Introduction*
- *Project Objective*
- *Working Principle of VGA display*
- *Implementation*
  - *Procedure*
  - *Flowchart*
  - *Module*
  - *Verilog Code*
- *Screenshots*
- *Future Scope*
- *Conclusion*
- *References*

Introduction

The idea of the project is to implement a 'printing and moving characters display' program in a VGA display using FPGA board . The program has pushbutton switches as input and a VGA monitor as an output device. We can print and animate characters in VGA screen. We have used Verilog for implementing the project. The goal of the project is to have a clear understanding of the vga monitor display using FPGA starting from specification to implementation.

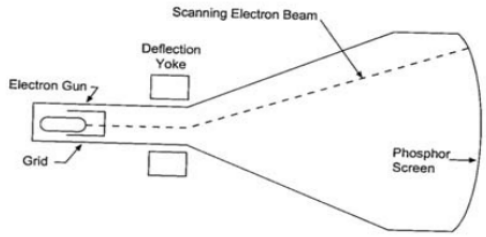
Project Objective

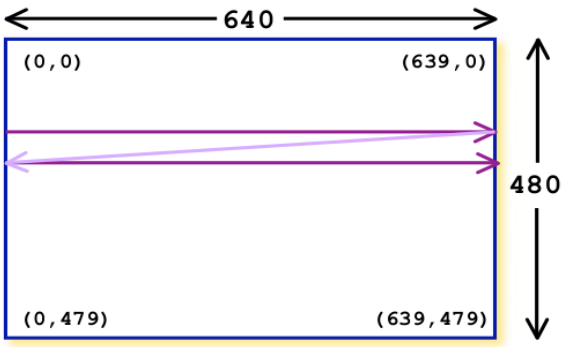
The objective of the project is to implement a program 'Print IIEST' in a VGA display using FPGA board . The program has pushbutton switches as input and a VGA monitor as an output device. We can print and animate IIEST,SHIBPUR character strings in VGA screen using this project. We have used Verilog for implementing the project. We also implement many formats of animating the string in this project. The project has the flexibility to change the starting and ending position of Strings , the size of the string and also the speed of movement of the strings.We have successfully implement all these things and the project is able to perform well.

Working Principle of VGA Display

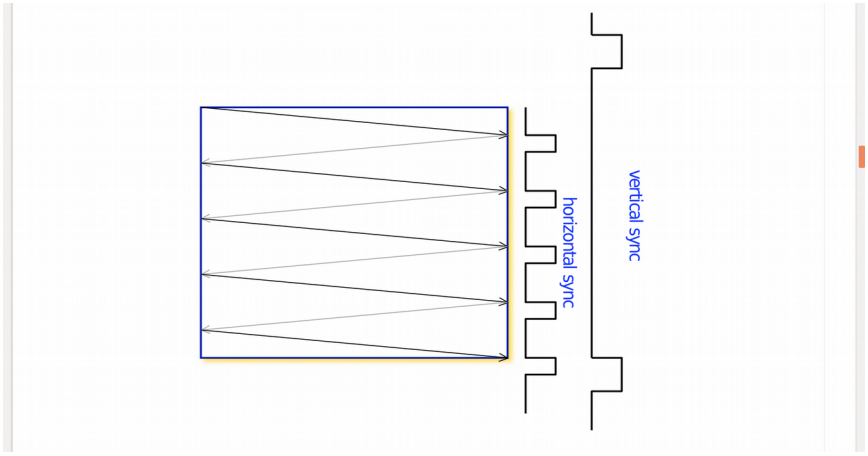
VGA video signal generation

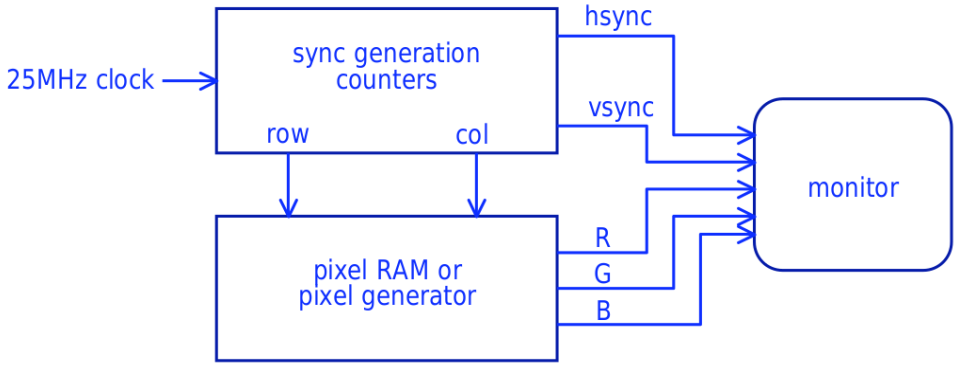
- A VGA video signal contains 5 active signals:
- *horizontal sync: digital signal, used for synchronisation of the video*
- *vertical sync: digital signal, used for synchronisation of the video*
- *red (R): analog signal (0-0.7 v), used to control the color*
- *green (G): analog signal (0-0.7 v), used to control the color*
- *blue (B): analog signal (0-0.7 v), used to control the color*
- *By changing the analog levels of the three RGB signals all other colors are produced*
- *The electron beam must be scanned over the viewing screen in a sequence of horizontal lines to generate an image. The RGB color information in the video signal is used to control the strength of the electron beam*
- *The screen refresh process begins in the top left corner and paints 1 pixel at a time from left to right. At the end of the first row, the row increments and the column address is reset to the first column. Once the entire screen has been painted, the refresh process begins again*
- *The video signal must redraw the entire screen 60 times per second to provide for motion in the image and to reduce flicker: this period is called the refresh rate. Refresh rates higher than 60 Hz are used in PC monitors*





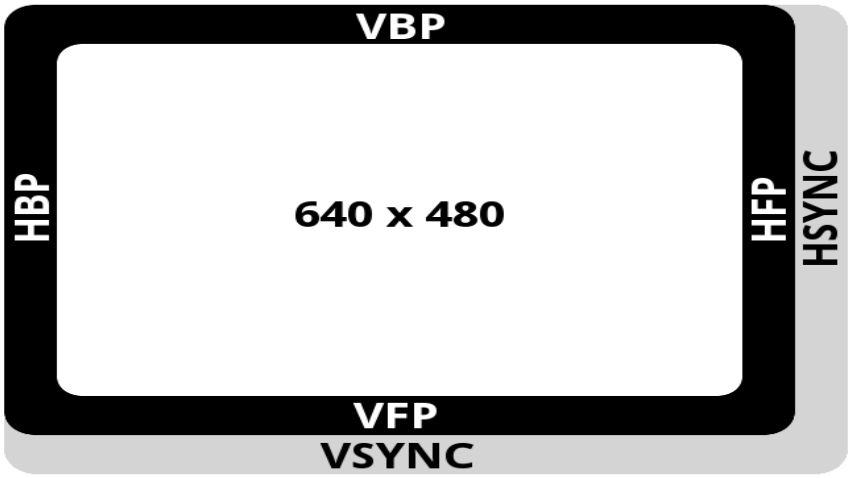
- *In 640 by 480-pixel mode, with a 60 Hz refresh rate, this is approximately 40 ns per pixel. A 25 MHz clock has a period of 40 ns*
- *The vertical sync signal tells the monitor to start displaying a new image or frame, and the monitor starts in the upper left corner with pixel (0,0)*
- *The horizontal sync signal tells the monitor to refresh another row of 640 pixels*
- *After 480 rows of pixels are refreshed with 480 horizontal sync signals, a vertical sync signal resets the monitor to the upper left corner and the process continues*
- *During the time when pixel data is not being displayed and the beam is returning to the left column to start another horizontal scan, the RGB signals should all be set to black color (all zero)*
- *In a PC graphics card, a dedicated memory location is used to store the color value of every pixel in the display. This memory is read out as the beam scans across the screen to produce the RGB signals*





Timing

Pixels on the screen are drawn in sequence, one by one. Rows are arranged top to bottom, and columns go from left to right. The row and column addresses are constantly incremented thus changing the position of currently drawn pixel. Synchronisation signals are used to tell the monitor to return the pixel back to the first row (VSYNC) or the first column (HSYNC).
Visible part of the screen is shown as the white area. It has a resolution of 640 by 480 pixels. Active video. This is the visible part of the screen, video output is enabled.



- *Front porch : When the trace reaches the end of the visible part of the screen, the video output is disabled. These areas are denoted as VPF (Vertical Front Porch) and HPF (Horizontal Front Porch) .*
- *Sync pulse : In case of HSYNC, the trace goes back to column zero. If pulse is VSYNC, the trace goes back to row zero. This part is also known as the retrace period.*
- *Back porch : This is the part that goes before the active video starts. These areas are denoted as VBF (Vertical Back Porch) and HBF (Horizontal Back Porch).*

25 MHz Pixel Clock	Active Video	Front Porch	Sync Pulse	Back Porch
Horizontal	640	16	96	48
Vertical	480	11	2	31

Implementation

Procedure :

- *To implement "Print IIEST" in the 'Vga display using Fpga' project first we have to go through the Vga monitor working principles, that is how the Vga works.*

- *Then we use Fpga(Field Programmable Gate Array) board and use the Vga interface of the fpga board and connect it with a Vga monitor .*

- *To print IIEST in Vga display we first synchronize the fpga clock with the Vga refresh rate frequency . After that we introduce qh,vh to actually calculate the visual active area of the Vga display and hcount,vcount to count the pixels in the Vga display active area.*

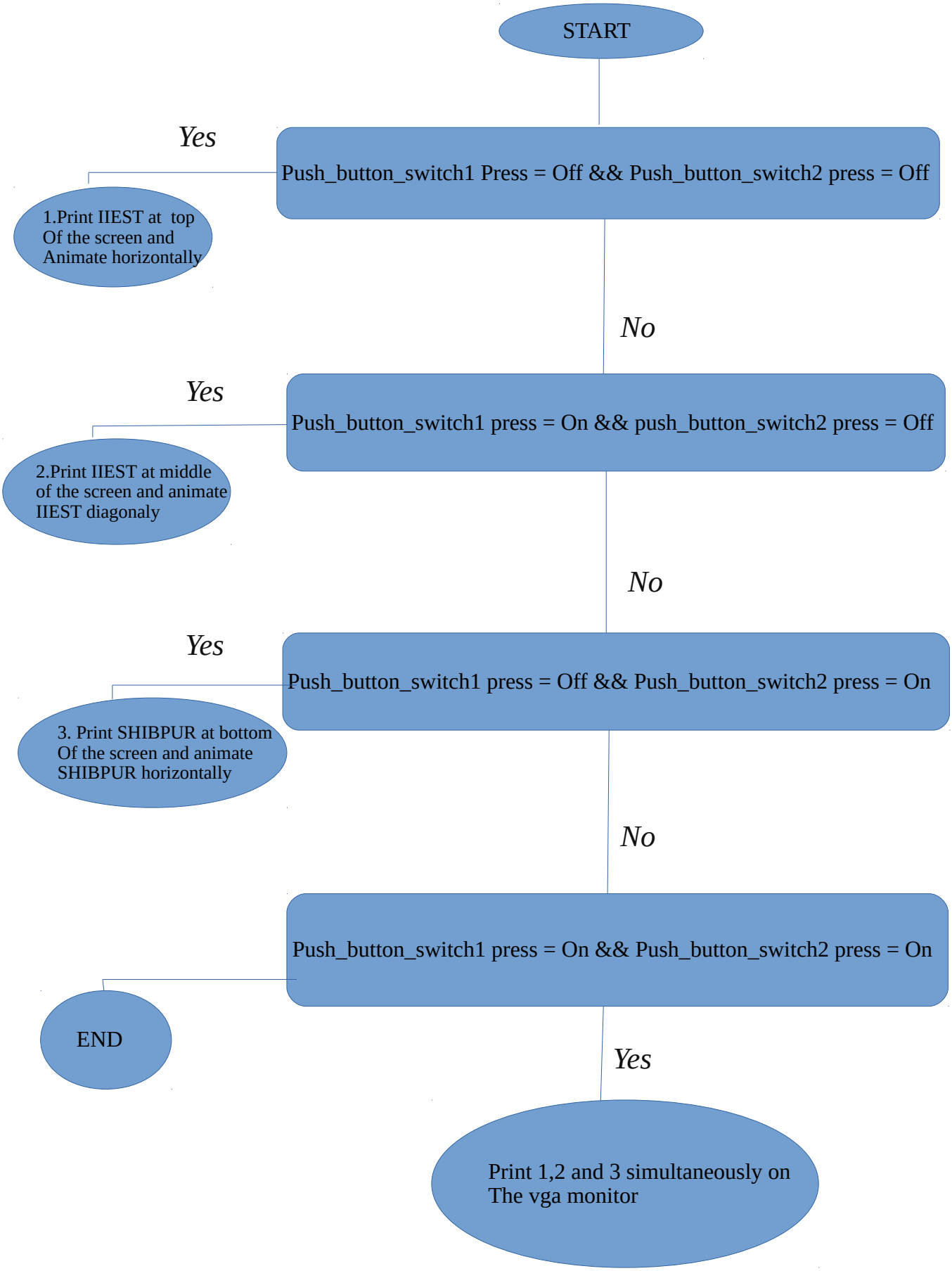
- *Then we use hc,vc,hc1,vc1,vc2 to slide charecters and the calculations of shifting characters are done with this variable under the frequency of slwclk .*

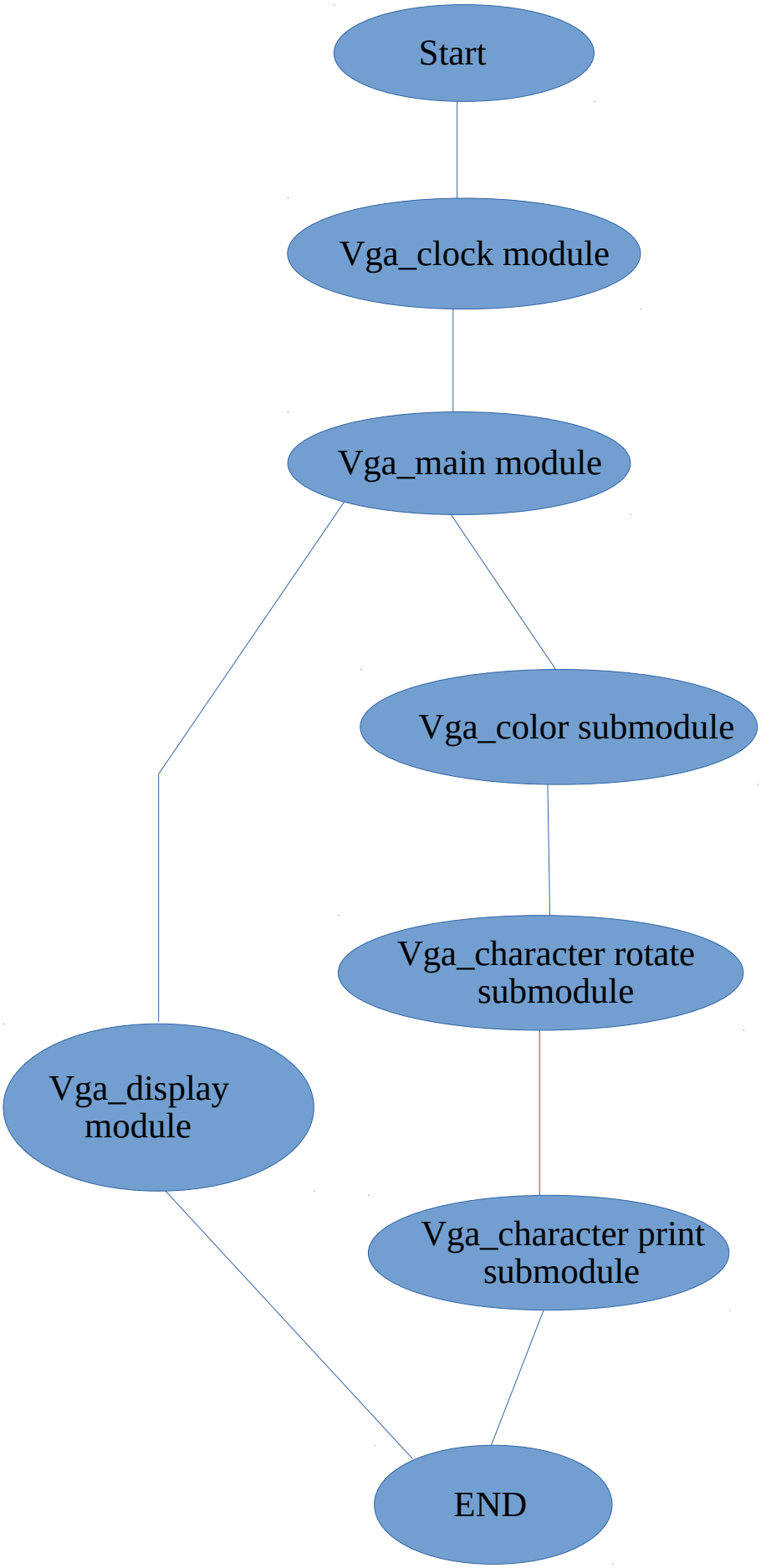
- *Then we assigns colours for the pixels where the characters are printed and rest pixels are remained black.The character colours are changed with time as we use colour array 'color[clr]' where we store the colours and each time there is a shift in characters we increment the value of the index of the colour array 'clr' and so the colours change with time.*

- *Then we use hc11,hc12,hc13,hc14,hc15,hc16,hc17 as a fuction of hc1 which are nothing but to calculate each character of 'IIEST' and 'SHIBPUR' separately as this two string rotate through the vga screen means each character of these two string vanishes away at the right hand side of the screen and re-occur at the left hand side of the screen,this movement are all happen smoothly and synchronously.*

- *And finally we write the code of how to print character string in Vga display without using Bitmap kind of concept.*

Flowcharts :


START
Push_button_switch1 Press = Off && Push_button_switch2 press = Off
Yes
1.Print IIEST at top Of the screen and Animate horizontally
No
Push_button_switch1 press = On && push_button_switch2 press = Off
Yes
2.Print IIEST at middle of the screen and animate IIEST diagonaly
No
Push_button_switch1 press = Off && Push_button_switch2 press = On
Yes
3. Print SHIBPUR at bottom Of the screen and animate SHIBPUR horizontally
No
Push_button_switch1 press = On && Push_button_switch2 press = On
END
Yes
Print 1,2 and 3 simultaneously on The vga monitor


Start
Vga_clock module
Vga_main module
Vga_color submodule
Vga_character rotate submodule
Vga_display module
Vga_character print submodule
END

Modules :

1. Vga clock module : This module slowdown the clock frequency of Fpga inbuilt clock(100 MHz) and matches with the clock frequency of the Vga monitor(25 Mhz).This module calls the Vga_main module.

2. Vga main module : This module determines the active display area of the Vga monitor . This module implements a slowclk which determines the frequency of the character movement in Vga monitor . This module gives the mechanism behind the characters movement across the Vga screen . This module calls the Vga_display module.

3. Vga display module : This module divides into three submodules.

- *Vga_color submodule : This module consists of the colour information which are used to colour charecters. We used 10 colours.*

- *Vga character rotate submodule : This module is behind the mechanism that is the characters are vanish away at the right hand side of the screen and re-occur themselves at the left hand side of the screen by matching with the normal characters movement speed and without making any delay at all . This appears as if that the characters are rotate along the screen.*

- *Vga character print submodule : This module prints the intended characters in the vga monitor and form words with the characters by making the spacing between the characters fixed . By this module we actually print 'IIEST' , 'SHIBPUR' in the vga monitor . We also make use of pushbutton switches to actually switch between the different visual printing formats that we employ in this project.*

Verilog Code :

iiest_print.v

Vga_clock module :

```
module printing_IIEST(mclk,rst,switch0,switch1,c,hs,vs);
        input mclk,rst;
        input switch0,switch1;
        output[7:0] c;
        output hs,vs;
        reg clk = 0;
        reg clk1 = 0;
        always @(posedge mclk)
        begin
              clk1 = ~clk1;
        end
        always @(posedge clk1)
        begin
                clk = ~clk;
        end
        main h_counter(clk,rst,switch0,switch1,hs,vs,c);
endmodule
```

Vga_main module :

```
module main(clk,rst,switch0,switch1,qh,vh,c);
        input clk,rst;
        input switch0,switch1;
        output reg qh;
        output reg vh;
        output wire[7:0] c;
```

```
reg[9:0] hcnt = 0;
reg[9:0] vcnt = 0;
reg[22:0] slwclk = 0;
wire slowclk;
integer hc=150,vc=100;
integer hc1 = 140,vc1 = 40;
integer vc2 = 460;
reg[2:0] clr = 3'b000;
always@(posedge clk)
begin
        slwclk = slwclk + 1;
end
assign slowclk = slwclk[22];
always @(posedge clk)
begin
        if(hcnt<799 && hcnt >95)
                qh <= 1;
        else
                qh <= 0;
        if(hcnt < 799)
                hcnt <= hcnt + 1;
        else
                hcnt <= 0;
        if(vcnt< 524 && vcnt >= 2)
                vh <= 1;
        else
                vh <= 0;
        if(hcnt == 799 && vcnt<524)
                vcnt <= vcnt + 1;
```

```
        if(vcnt > 523)
                vcnt <= 0;
end
always@(posedge slowclk)
begin
        if(hc <601 && vc <381)
                begin
                        hc = hc +15;
                        vc = vc + 10;
                        if(clr == 3'b111)
                                clr = 3'b000;
                        else
                                clr = clr + 3'b001;
                end
else
                begin
                        hc = 160;
                        vc = 110;
                        if(clr == 3'b111)
                                clr = 3'b000;
                        else
                                clr = clr + 3'b001;
                end
end
always@(posedge slowclk)
begin
        hc1 = (hc1+10);
        if(hc1 == 780)
                hc1 = 140;
```

```
        end
        printI iprint1(hcnt,vcnt,hc,vc,hc1,vc1,vc2,clr,switch0,switch1,c);
endmodule
```

Vga_display module:

```
module printI(hcnt,vcnt,hc,vc,hc1,vc1,vc2,clr,switch0,switch1,c);
                        input [9:0] hcnt;
                        input [9:0] vcnt;
                        input [9:0] hc;
                        input [9:0] vc;
                        input [9:0] hc1;
                        input [9:0] vc1;
                        input [9:0] vc2;
                        input  [2:0] clr;
                        input switch0,switch1;
                        reg[7:0] color[0:9];
                        integer hc11,hc12,hc13,hc14,hc15,hc16,hc17;
                        output reg[7:0] c;
```

Vga_colour submodule :

```
                        initial
                        begin
                        color[9] <= 8'b11111111;
                        color[8] <= 8'b10010011;
                        color[7] <= 8'b11011001;
                        color[6] <= 8'b10010111;
                        color[5] <= 8'b00011111;
                        color[4] <= 8'b11100011;
                        color[3] <= 8'b11111100;
                        color[2] <= 8'b11000000;
                        color[1] <= 8'b00111000;
```

```
color[0] <= 8'b00000111;
end
```

Vga_character_rotate submodule:

```
always @(*)
begin
        hc11 = hc1 % 780;
        if(hc11 < 1)
            hc11 = hc11 +140;
        hc12 = (hc1 + 40) % 780;
        if(hc12 < 41)
            hc12 = hc12 +140;
        hc13 = (hc1 + 80) % 780;
        if(hc13 < 81)
            hc13 = hc13 + 140;
        hc14 = (hc1 + 120) % 780;
        if(hc14 < 121)
            hc14 = hc14 + 140;
        hc15 = (hc1 + 160) % 780;
        if(hc15 < 161)
            hc15 = hc15 + 140;
        hc16 = (hc1 + 200) % 780;
        if(hc16 < 201)
            hc16 = hc16 + 140;
        hc17 = (hc1 + 240) % 780;
        if(hc17 < 241)
            hc17 = hc17 + 140;
    end
```

Vga_character_print submodule :

```
if((vcnt>= vc1 && vcnt<= vc1 + 10'd10) && ((switch0 == 1 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))
```

```
                if((hcnt>hc11 && hcnt<(hc11 + 10'd30) )||(hcnt>(hc12) &&
hcnt<(hc12 +10'd30) ) ||(hcnt>(hc13) && hcnt<(hc13+10'd30) )||(hcnt>(hc14) &&
hcnt<(hc14 +10'd30) ) ||(hcnt > (hc15) && hcnt < (hc15 +10'd30) ))

                        c = color[clr];

                        else

                        c = 8'b00000000;

            else if((vcnt>= vc1+ 10'd10 && vcnt<= vc1 + 10'd20) && ((switch0 ==
1 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

            if((hcnt>(hc11 + 10'd10) && hcnt<(hc11 + 10'd20) )||(hcnt>(hc12 +
10'd10) && hcnt<(hc12 + 10'd20) )||(hcnt>(hc13) && hcnt<(hc13 + 10'd10) )||
(hcnt>(hc14) && hcnt<(hc14 + 10'd10) )||(hcnt>(hc15 + 10'd10) && hcnt<(hc15 +
10'd20) ))

                        c = color[clr];

                        else

                        c = 8'b00000000;

            else if((vcnt>= vc1 + 10'd20 && vcnt<= vc1 + 10'd30) && ((switch0 ==
1 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

            if((hcnt>(hc11 + 10'd10) && hcnt<(hc11 + 10'd20) )||(hcnt>(hc12 +
10'd10) && hcnt<(hc12 + 10'd20) )||(hcnt>(hc13)  && hcnt<(hc13 + 10'd30) )||
(hcnt>(hc14 ) && hcnt<(hc14 + 10'd30) )||(hcnt>(hc15 + 10'd10) && hcnt<(hc15 +
10'd20) ))

                        c = color[clr];

                        else

                        c = 8'b00000000;

            else if((vcnt>= vc1 + 10'd30 && vcnt<= vc1 + 10'd40) && ((switch0 ==
1 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

            if((hcnt>(hc11 + 10'd10) && hcnt<(hc11 + 10'd20) )||(hcnt>(hc12 +
10'd10) && hcnt<(hc12 + 10'd20) )||(hcnt>(hc13)  && hcnt<(hc13 + 10'd10) )||
(hcnt>(hc14 + 10'd20) && hcnt<(hc14 + 10'd30) )||(hcnt>(hc15 + 10'd10) &&
hcnt<(hc15 + 10'd20) ))

                        c = color[clr];

                        else

                        c = 8'b00000000;

           else if((vcnt>= vc1+ 10'd40 && vcnt<= vc1 + 10'd50) && ((switch0 == 1
&& switch1 == 1) || (switch0 == 0 && switch1 == 0)))
```

```
                if((hcnt>hc11 && hcnt<(hc11 + 10'd30) )||(hcnt>(hc12 ) && hcnt<(hc12
+10'd30) )||(hcnt>(hc13) && hcnt<(hc13 +10'd30) )||(hcnt>(hc14) && hcnt<(hc14
+10'd30) )||(hcnt>(hc15 + 10'd10) && hcnt<(hc15 +10'd20) ))

                                    c = color[clr];

                                    else

                                    c = 8'b00000000;


                 else if((vcnt>vc && vcnt<= vc + 10'd10) && ((switch0 == 1 && switch1
== 0) || (switch0 == 0 && switch1 == 0)))

                 if((hcnt>hc && hcnt<hc + 10'd30)||(hcnt>hc + 10'd40 && hcnt<hc
+10'd70)||(hcnt>hc +10'd80 && hcnt<hc +10'd110)||(hcnt>hc +10'd120 && hcnt<hc
+10'd150)||(hcnt>hc +10'd160 && hcnt<hc +10'd190))

                                    c = color[clr];

                                    else

                                    c = 8'b00000000;

                else if((vcnt>vc + 10'd10 && vcnt<=vc + 10'd20) && ((switch0 == 1 &&
switch1 == 0) || (switch0 == 0 && switch1 == 0)))

                 if((hcnt>hc + 10'd10 && hcnt<hc + 10'd20)||(hcnt>hc + 10'd50 &&
hcnt<hc + 10'd60)||(hcnt>hc + 10'd80 && hcnt<hc + 10'd90)||(hcnt>hc + 10'd120
&& hcnt<hc + 10'd130)||(hcnt>hc + 10'd170 && hcnt<hc + 10'd180))

                                    c = color[clr];

                                    else

                                     c = 8'b00000000;

                  else if((vcnt>vc + 10'd20 && vcnt<=vc + 10'd30)&& ((switch0 == 1 &&
switch1 == 0) || (switch0 == 0 && switch1 == 0)))

                   if((hcnt>hc + 10'd10 && hcnt<hc + 10'd20)||(hcnt>hc + 10'd50 &&
hcnt<hc + 10'd60)||(hcnt>hc + 10'd80 && hcnt<hc + 10'd110)||(hcnt>hc + 10'd120
&& hcnt<hc + 10'd150)||(hcnt>hc + 10'd170 && hcnt<hc + 10'd180))

                                     c = color[clr];

                                    else

                                    c = 8'b00000000;

                    else if((vcnt>vc +10'd30 && vcnt<=vc + 10'd40)&& ((switch0 == 1 &&
switch1 == 0) || (switch0 == 0 && switch1 == 0)))

                     if((hcnt>hc + 10'd10 && hcnt<hc + 10'd20)||(hcnt>hc + 10'd50 &&
hcnt<hc + 10'd60)||(hcnt>hc + 10'd80 && hcnt<hc + 10'd90)||(hcnt>hc + 10'd140
```

```
&& hcnt<hc + 10'd150)||(hcnt>hc + 10'd170 && hcnt<hc + 10'd180))

                                c = color[clr];

                                else

                                c = 8'b00000000;


                else if((vcnt>vc +10'd40 && vcnt<= vc + 10'd50)&& ((switch0 == 1 && switch1 == 0) || (switch0 == 0 && switch1 == 0)))

                if((hcnt>hc && hcnt<hc + 10'd30)||(hcnt>hc + 10'd40 && hcnt<hc +10'd70)||(hcnt>hc +10'd80 && hcnt<hc +10'd110)||(hcnt>hc +10'd120 && hcnt<hc +10'd150)||(hcnt>hc + 10'd170 && hcnt<hc +10'd180))

                                c = color[clr];

                                else

                                c = 8'b00000000;

                 else if((vcnt>vc2 && vcnt<= vc2 + 10'd10) && ((switch0 == 0 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

                if((hcnt>hc11 && hcnt<hc11 + 10'd30)||(hcnt>hc12 && hcnt<hc12 +10'd10)||(hcnt>hc12 + 10'd20 && hcnt<hc12 +10'd30)||(hcnt>hc13 && hcnt<hc13 +10'd30)||(hcnt>hc14 && hcnt<hc14 +10'd30)||(hcnt>hc15 && hcnt<hc15 +10'd30)||(hcnt>hc16 && hcnt<hc16 +10'd10)||(hcnt>hc16 +10'd20 && hcnt<hc16 +10'd30)||(hcnt>hc17 && hcnt<hc17 +10'd30))

                                c = color[clr];

                                else

                                c = 8'b00000000;

                else if((vcnt>vc2 + 10'd10 && vcnt<=vc2 + 10'd20) && ((switch0 == 0 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

                if((hcnt>hc11 && hcnt<hc11 + 10'd10)||(hcnt>hc12 && hcnt<hc12 +10'd10)||(hcnt>hc12 + 10'd20 && hcnt<hc12 +10'd30)||(hcnt>hc13 +10'd10 && hcnt<hc13 +10'd20)||(hcnt>hc14 && hcnt<hc14 +10'd10)||(hcnt>hc14 +10'd20 && hcnt<hc14 +10'd30)||(hcnt>hc15 && hcnt<hc15 +10'd10)||(hcnt>hc15 +10'd20 && hcnt<hc15 +10'd30)||(hcnt>hc16 && hcnt<hc16 +10'd10)||(hcnt>hc16 +10'd20 && hcnt<hc16 +10'd30)||(hcnt>hc17 && hcnt<hc17 +10'd10)||(hcnt>hc17 +10'd20 && hcnt<hc17 +10'd30))

                                c = color[clr];

                                else

                                c = 8'b00000000;
```

```
		else if((vcnt>vc2 + 10'd20 && vcnt<=vc2 + 10'd30) && ((switch0 == 0 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

		if((hcnt>hc11 && hcnt<hc11 + 10'd30)||(hcnt>hc12 && hcnt<hc12 +10'd30)||(hcnt>hc13 + 10'd10 && hcnt<hc13 +10'd20)||(hcnt>hc14  && hcnt<hc14 +10'd25)||(hcnt>hc15  && hcnt<hc15 +10'd30)||(hcnt>hc16  && hcnt<hc16 +10'd10)||(hcnt>hc16 +10'd20 && hcnt<hc16 +10'd30)||(hcnt>hc17 && hcnt<hc17 +10'd25))

					c = color[clr];

					else

					c = 8'b00000000;

		else if((vcnt>vc2 +10'd30 && vcnt<=vc2 + 10'd40) && ((switch0 == 0 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

		if((hcnt>hc11 + 10'd20 && hcnt<hc11 + 10'd30)||(hcnt>hc12  && hcnt<hc12 +10'd10)||(hcnt>hc12 + 10'd20 && hcnt<hc12 +10'd30)||(hcnt>hc13 +10'd10 && hcnt<hc13 +10'd20)||(hcnt>hc14  && hcnt<hc14 +10'd10)||(hcnt>hc14 +10'd20 && hcnt<hc14 +10'd30)||(hcnt>hc15  && hcnt<hc15 +10'd10)||(hcnt>hc16 && hcnt<hc16 +10'd10)||(hcnt>hc16 +10'd20 && hcnt<hc16 +10'd30)||(hcnt>hc17 && hcnt<hc17 +10'd10)||(hcnt>hc17 +10'd20 && hcnt<hc17 +10'd30))

					c = color[clr];

					else

					c = 8'b00000000;

		else if((vcnt>vc2 +10'd40 && vcnt<= vc2 + 10'd50) && ((switch0 == 0 && switch1 == 1) || (switch0 == 0 && switch1 == 0)))

		if((hcnt>hc11 && hcnt<hc11 + 10'd30)||(hcnt>hc12  && hcnt<hc12 +10'd10)||(hcnt>hc12 + 10'd20 && hcnt<hc12 +10'd30)||(hcnt>hc13  && hcnt<hc13 +10'd30)||(hcnt>hc14  && hcnt<hc14 +10'd30)||(hcnt>hc15 && hcnt<hc15 +10'd10)||(hcnt>hc16  && hcnt<hc16 +10'd30)||(hcnt>hc17 && hcnt<hc17 +10'd10)||(hcnt>hc17 +10'd20 && hcnt<hc17 +10'd30))

					 c = color[clr];

					 else

					c = 8'b00000000;

		 else

		  c=8'b00000000;

	end

endmodule
```

iiest_print.ucf

NET "mclk" LOC=V10 | IOSTANDARD=LVCMOS33 | PERIOD=100MHz ;

NET "rst" LOC=T6 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST | PULLUP ;

NET "switch0" LOC=F14 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST | PULLUP ;

NET "switch1" LOC=A14 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST | PULLUP ;

NET "hs" LOC=U8 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "vs" LOC=V8 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[0]" LOC=T11 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[1]" LOC=R11 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[2]" LOC=M8 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[3]" LOC=N8 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

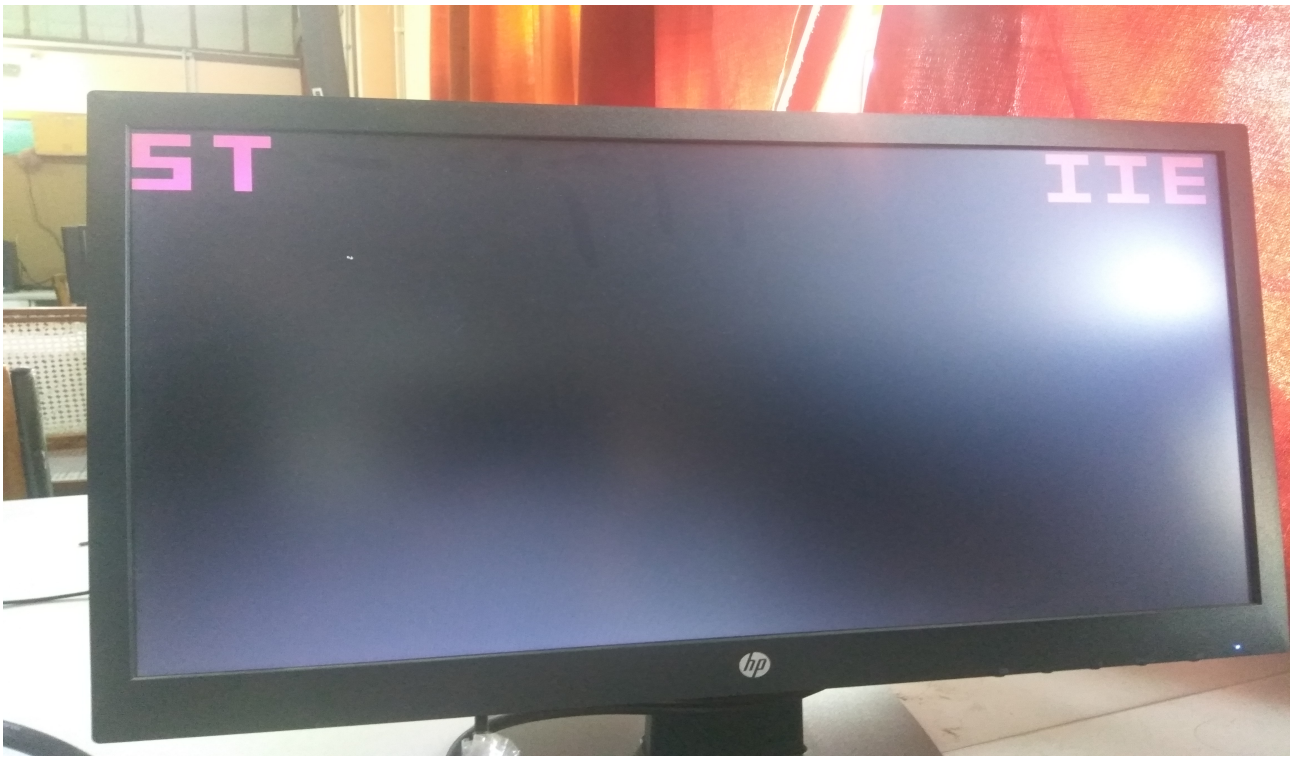
NET "c[4]" LOC=P8 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[5]" LOC=N7 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

NET "c[6]" LOC=N6 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

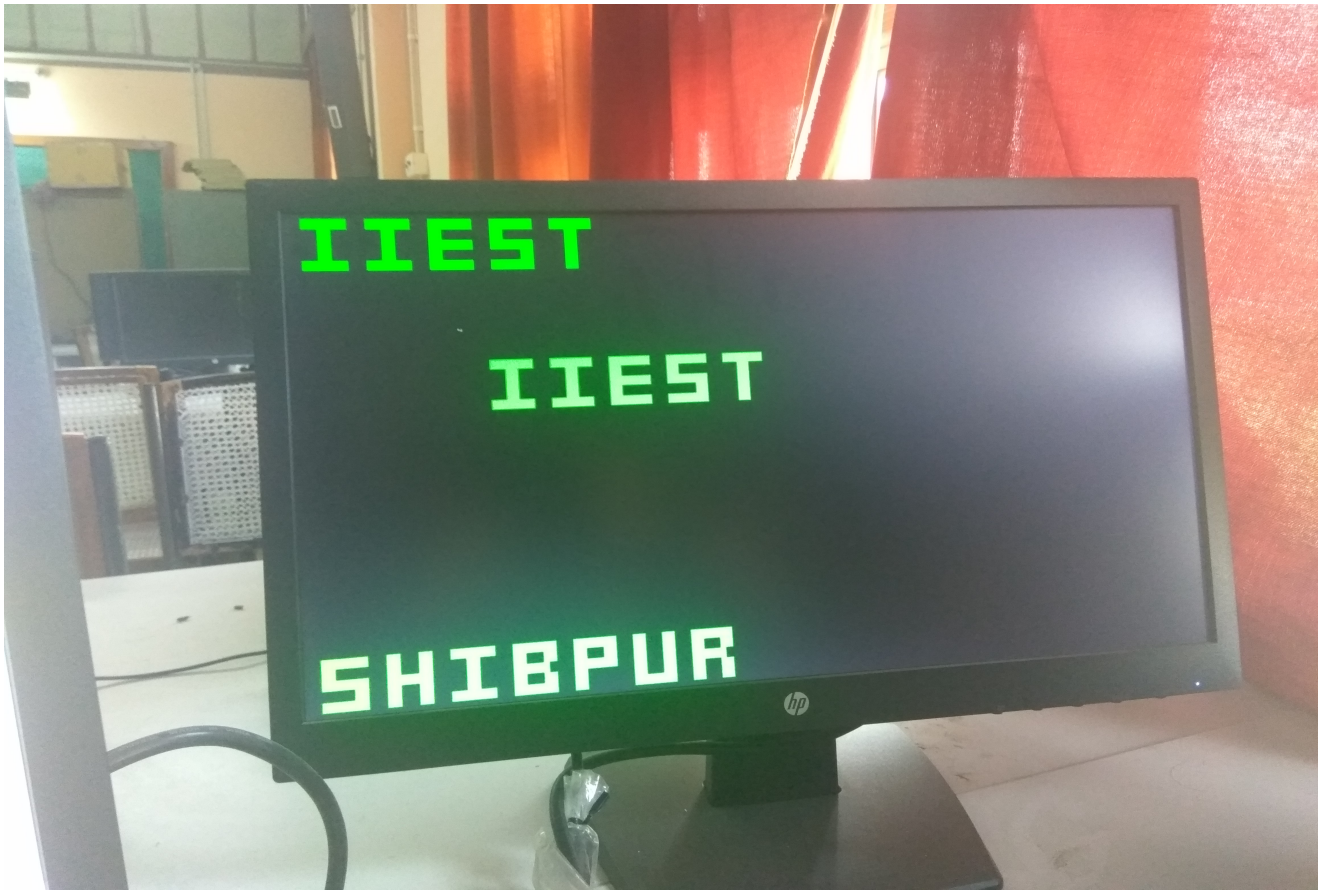
NET "c[7]" LOC=P7 | IOSTANDARD=LVCMOS33 | DRIVE=8 | SLEW=FAST ;

Screenshots




SHIBPUR


R
SHIBPU


IIEST
IIEST
SHIBPUR


ST
IIE
IIEST
BPUR
SHI

Future Scope

This project can be used as a tool to print any string and animate that string in any format in a vga monitor . In future if we implement bitmap in this project then it can be more easier to generate any string to view on a vga monitor and it will be very useful to simulate it as a one type of animation software.

Conclusion

The project offered us a great chance to have practical experience on Vga monitor display design and achieved working level expertise on Vga and related areas. We had to use hardware components like Fpga board and Vga monitor as well as software parts like verilog code , Xilinx platform and integrate between them. Also throughout the project we had to study extensively on related matters while working on the project. That greatly improved our skills of approaching towards a problem and solving it. We gained a lot of knowledge from related things while working that might not be connected with the current project but contributed a lot for our future works in this area. Debugging an system requires a lot of patience and is very time consuming and it can be very frustrating when you encounter problem but it was very satisfying and rewarding both mentally and educationally to manage building something from scratch.

References

1. https://embeddedthoughts.com/2016/07/29/driving-a-vga-monitor-using-an-fpga/

2. wikipedia

3. Youtube

4. https://www.fpga4fun.com/PongGame.html

5. Github