# CS207 Digital Logic: Final Project Report

**Group Members**:

12013027 YU Kunyi 余坤屹

12011905 GAO Jianxiong 高健雄

**Project Selection**:

Project ONE (Simple VGA & Tangram)

**Report content**:



# 1 Exploitation arrangement & implementation logs

## 1.1 Schedule & implementation

Week 8: (Apr. 6th) Confirm groupmate

Week 9: (Apr. 11th) Choose the project topic 1

Week 10 & 11: Learn the basic principles and theories through

(1) The textbooks[1] [2] [3] borrowed from the library;

(2) Webs of "设计说明与提示" in guide pdf;

(3) Others webs like GitHub and CSDN.

Week 12: null

Week 13: Begin to design the very first edition and complete the task "stripes". Display unsuccessful : (

Week 14: Modify the VGA_800x600.v and finally display the figure "stripes" and multiple geometric figure.

Week 15: Code the move and rotation but still has bug.

Week 16: Debug and finish. Write the report and make video.

## 1.2 Tasks allocation

### (1) YU Kunyi 余坤屹

1. Overall programme structure designation;

2. Complete drawing the stripes and multiple geometric figure e.g. square, circle, triangle etc. ;

3. Code and debug module VGA_top.v, VGA_800x600.v, VGA_display_v2.v (and its 4 sub-module), ip-core clock div;

4. Do presentation and report writing.

### (2) GAO Jianxiong 高健雄

1. Searching for the information from the point of principle and advise for debug;

2. Code and debug module Binary_To_7Segments.v and LED;

3. Make slides and do presentation.

### (3) Workload ratio

余坤屹 60%

高健雄 40%

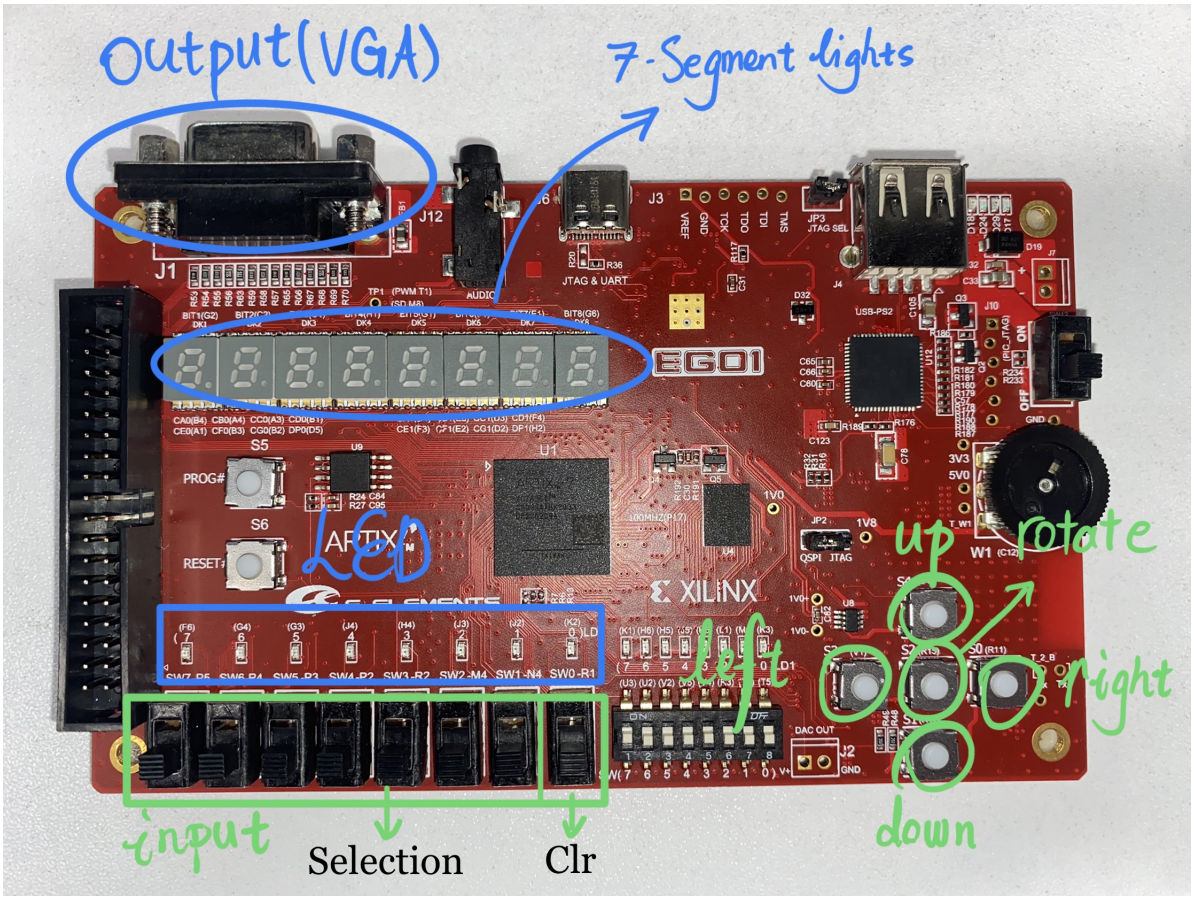
# 2 Designation

## 2.1 Demand analysis

### (1) System function:

Using the VGA interface on EGO1 board to draw and control (move and rotate) various geometric shapes, and finally make a pair of tangram.

### (2) Input devices & (3) Output devices:



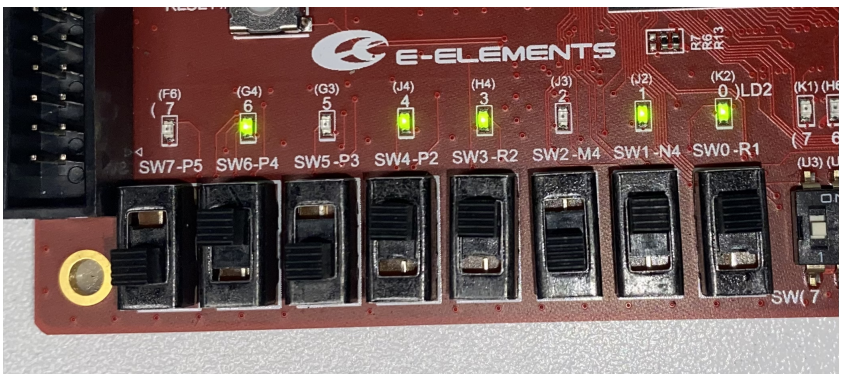

## (4) Interface standard:

Output:

(1) VGA port obey the industrial standard VGA 800x600:

| Format | Pixel Clock (MHz) | Horizontal (in Pixels) | | | | Vertical (in Lines) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Active Video | Front Porch | Sync Pulse | Back Porch | Active Video | Front Porch | Sync Pulse | Back Porch |
| 800x600, 60Hz | 40.000 | 800 | 40 | 128 | 88 | 600 | 1 | 4 | 23 |

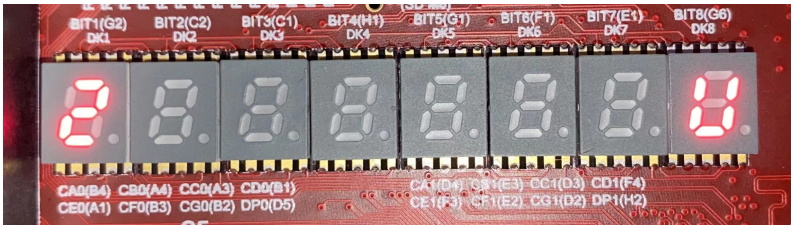
(2) The LED lights up when the DIP switch is up;




(3) 7-segment lights:

We use the leftmost one to show a number 0-7 to denote the chosen geometric figure (according to the DIP switch up-and-down).

And use the rightmost one to show the type of current operation (move or rotate). In details: (1) "U" means upwards; (2) "D" means downwards; (3) "L" means leftwards; (4) "R" means rightwards; (5) "S" means rotation or spinning.




Input:

(1) Clr:

It means "clear", using the rightmost DIP switch "SW0-R1". Function: activate or refresh the screen. (It is not the reset)

(2) Selection:

Using left 7 DIP switch to choose the geometric figure in screen you want to control. Eventually, you could push upward multiple DIP switch to control them at the same time.

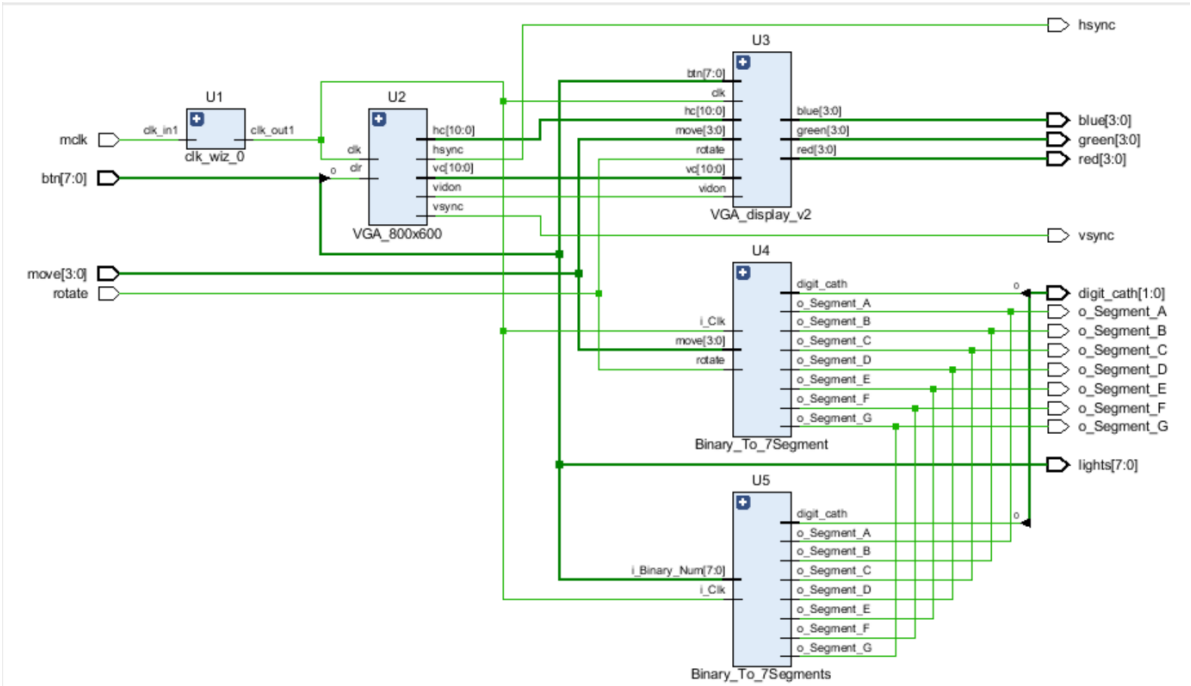
(3) Four movement control bottom:

Long press to move in a particular direction. If press multiple bottom at the same time, it will control according to the priority of up, down, left and right.

(4) Rotation bottom:

Long press to rotate. Rotate 90° per second.

## 2.2 System construction figure (RTL)




- The function you could find in below part : )

## 2.3 Module analysis

### TOP Module (VGA_top.v):

- **Function**: Implement all the sub-module and interact with hardware.
- **Code**:

```
`include "VGA_800x600.v"
`include "VGA_display_v2.v"
`include "Binary_To_7Segment.v"
`include "Binary_To_7Segments.v"

module VGA_top (input wire mclk,
                (* dont_touch = "true" *) input wire [7:0] btn,
                (* dont_touch = "true" *) input wire [3:0] move,
                input wire rotate,
                output wire hsync,
                vsync,
                output wire [3:0] red,
                green,
                blue,
                output wire [7:0] lights,
                output      o_Segment_A,
                output      o_Segment_B,
                output      o_Segment_C,
                output      o_Segment_D,
                output      o_Segment_E,
                output      o_Segment_F,
                output      o_Segment_G,
                output [1:0] digit_cath,
                output      Segment_A,
                output      Segment_B,
                output      Segment_C,
                output      Segment_D,
                output      Segment_E,
                output      Segment_F,
                output      Segment_G
                );

    wire clk_out, clr, vidon;
    wire [10:0] hc, vc;
    assign clr = btn[0];
    assign lights = btn;

    clk_wiz_0 U1(
    .clk_in1(mclk),//100MHz
    .clk_out1(clk_out)//40MHz
    );

    VGA_800x600 U2(
    .clk(clk_out),
    .clr(clr),
    .hsync(hsync),
    .vsync(vsync),
    .hc(hc),
    .vc(vc),
    .vidon(vidon)
```

```
    );

    VGA_display_v2 U3(
    .vidon(vidon),
    .hc(hc),
    .vc(vc),

    .red(red),
    .green(green),
    .blue(blue),

    .btn(btn),
    .move(move),
    .rotate(rotate),
    .clk(clk_out)
    );

    Binary_To_7Segment U4(
        .i_Clk(clk_out),
        .move(move),
        .rotate(rotate),
        .o_Segment_A(o_Segment_A),
        .o_Segment_B(o_Segment_B),
        .o_Segment_C(o_Segment_C),
        .o_Segment_D(o_Segment_D),
        .o_Segment_E(o_Segment_E),
        .o_Segment_F(o_Segment_F),
        .o_Segment_G(o_Segment_G),
        .digit_cath(digit_cath[0])
    );

    Binary_To_7Segments U5(
        .i_Clk(clk_out),
        .i_Binary_Num(btn),
        .Segment_A(Segment_A),
        .Segment_B(Segment_B),
        .Segment_C(Segment_C),
        .Segment_D(Segment_D),
        .Segment_E(Segment_E),
        .Segment_F(Segment_F),
        .Segment_G(Segment_G),
        .digit_cath(digit_cath[1])
    );
endmodule // VGA_top
```

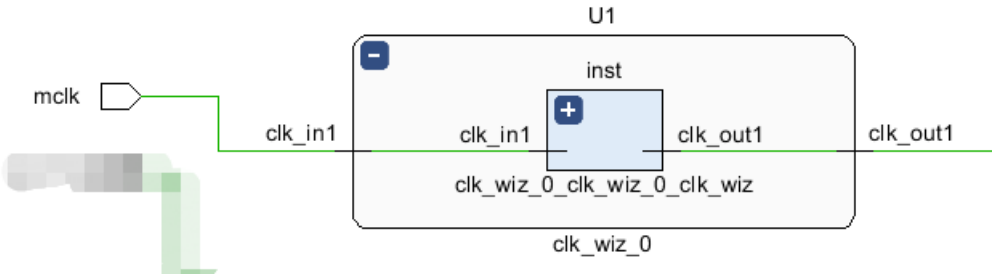
VGA_top.v

You can find U1 to U5 in VGA_top.v.

### Module U1 (clk_wiz_0):

- **Function**: Simple divide the 100MHz clock to output 40MHz.
- **I/O**: "clk_in1" and "clk_out1"

- **RTL**:




- **Code**:

```
clk_wiz_0 U1(
.clk_in1(mclk),
.clk_out1(clk_out)
);
```

in VGA_top.v

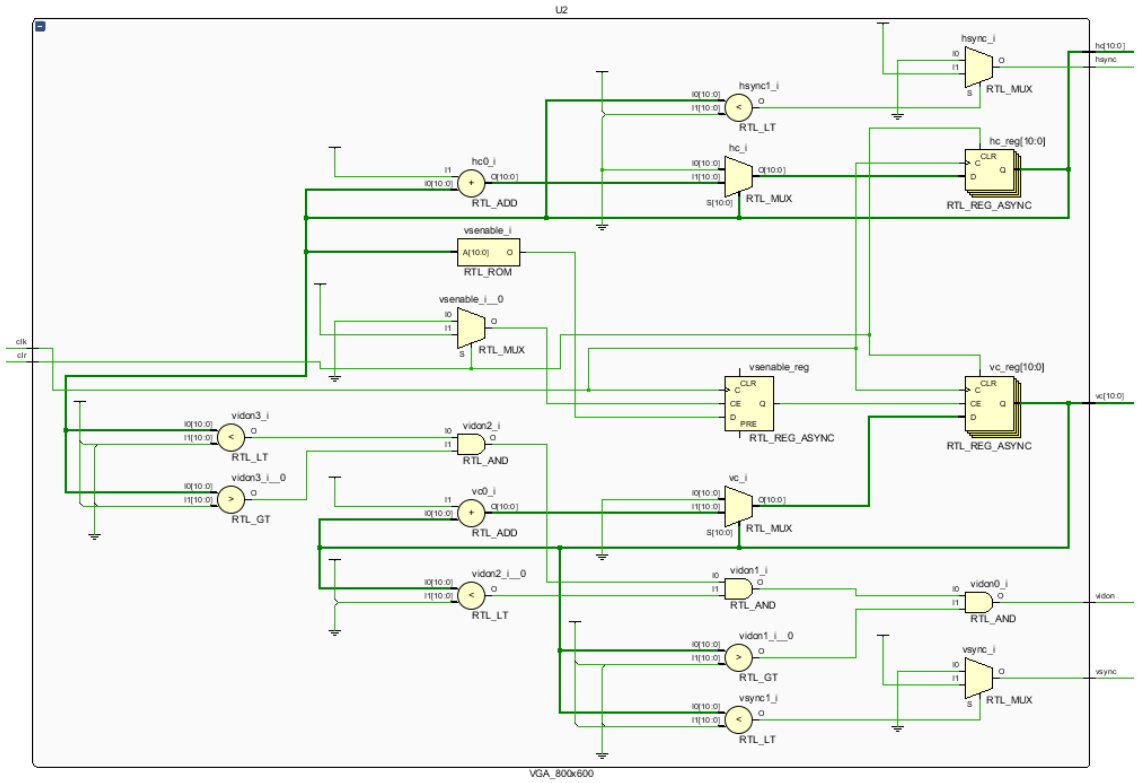
- **Attention**: The src files does not includes this part. You could new this module by (1) In Vivado -> IP Catalog -> search and click "Clock Wizard"; (2) Do not changing default name, change "Output Clocks" -> "clk_out1" -> "Output Freq(MHz)" = 40; (3) In this page, unable "reset" and "locked" (4) Generate.

## Module U2 (VGA_800x600):

- **Function**: Control the display in screen follow the standard of 800x600 60Hz.
- **I/O**: Input a 40MHz clock sign;

  output "hsync", "vsync", "hc", "vc", "vidon", the standard VGA 800x600 signs.

- **RTL**:




- **Code**:

```
VGA_800x600 U2(
    .clk(clk_out),
    .clr(clr),
    .hsync(hsync),
    .vsync(vsync),
    .hc(hc),
    .vc(vc),
    .vidon(vidon)
    );
```

in VGA_top.v

```
module VGA_800x600 (input wire clk,       // 时钟脉冲
                    input wire clr,       // 清零
                    output reg hsync,     // HSync 行同步信号
                    output reg vsync,     // VSync 场同步信号
                    output reg [10:0] hc, // count for HSync
                    output reg [10:0] vc, // count for VSync
                    output reg vidon);    // 1 for count is in visiable range

    parameter hpixels = 11'b10000100000; // 行像素点 = 1056
    parameter vlines  = 11'b01001110100; // 行数  = 628
    parameter hbp     = 11'b00011011000; // 行显示后沿     = 216
    parameter hfp     = 11'b01111111000; // 行显示前沿     = 1016
    parameter vbp     = 11'b00000011011; // 场显示后沿     = 27
    parameter vfp     = 11'b01001110011; // 场显示前沿     = 627
    reg vsenable      = 0; // Enable for the Vertical counter

    initial begin
        hsync = 0;
        vsync = 0;
        hc    = 0;
        vc    = 0;
        vidon = 0;
    end

    // 行同步信号计数器
    always @(posedge clk or posedge clr) begin
        if (clr == 1)
            hc <= 0;
        else begin
            if (hc == hpixels - 1) begin
                hc       <= 0;
                vsenable <= 1;
            end
            else begin
                hc       <= hc + 1;
                vsenable <= 0;
            end
        end
    end

    // 产生HSync脉冲
    // 当hc = 0~127时，行同步脉冲为低电ping
    always @ (*)
    begin
```

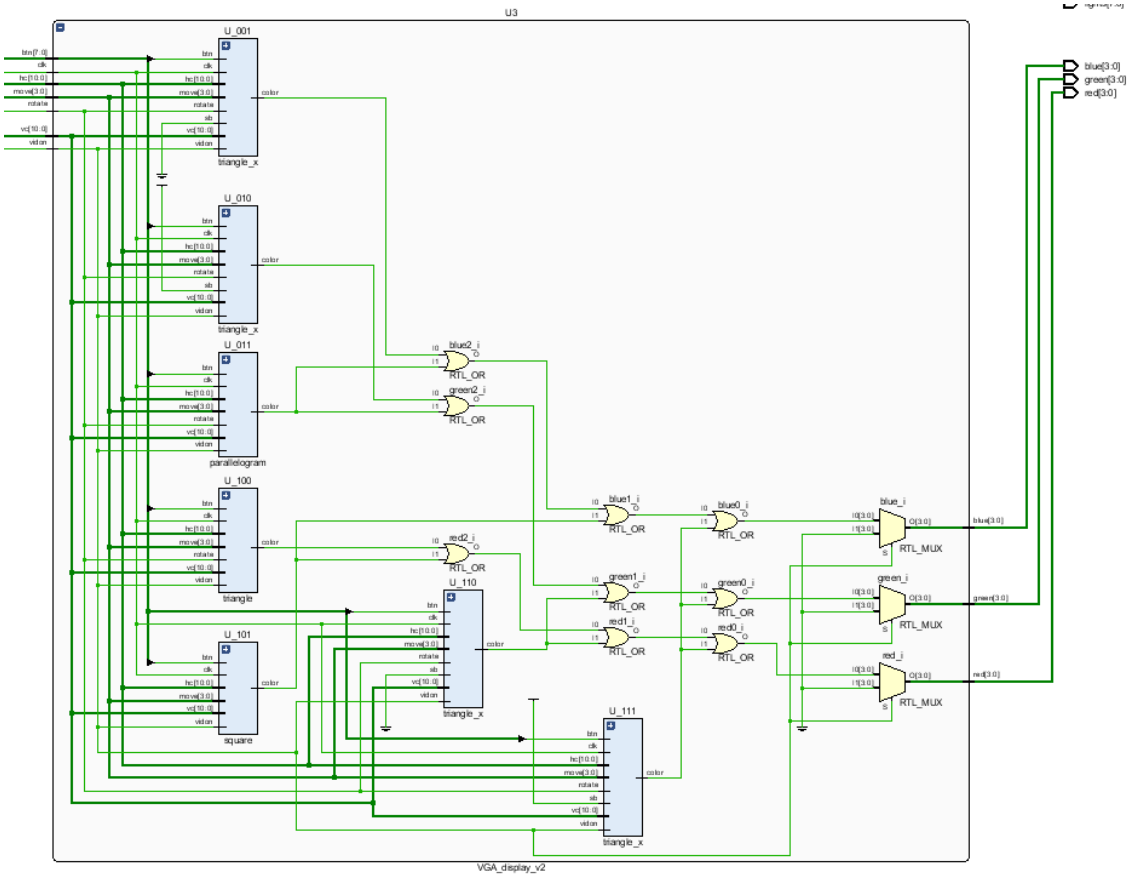
```
        if (hc < 128) hsync = 0;
        else hsync          = 1;
    end

    // 场同步信号计数器
    always @(posedge clk or posedge clr) begin
        if (clr == 1) vc <= 0;
        else begin
        if (vsenable == 1) begin
            if (vc == vlines - 1) vc <= 0;
            else vc <= vc + 1;
            end
        end
    end

    // 产生VSync脉冲
    // When hc = 0 or 1, 场同步脉冲为低电ping
    always @(*) begin
        if (vc < 4) vsync = 0;
        else vsync        = 1;
    end

    // Enable video out when within the proches
    always @ (*) begin
        if ((hc < hfp) && (hc > hbp) && (vc < vfp) && (vc > vbp)) begin
            vidon = 1;
        end
        else vidon = 0;
    end

endmodule //VGA_800x600
```

VGA_800x600.v

## Module U3 (VGA_display_v2):

- **Function**: According the input "hc" and "vc" (denote the position in screen), determine the color output. This module contains 7 sub-module with control 7 geometric figure of tangram.
- **I/O**: input "vidon", "hc", "vc", "[7:0] btn" (choose the control geometric figure), "[3:0] move", "rotate"

  output [3:0] "red", "green" and "blue".
- **RTL**:




- **Code**:

```
    VGA_display_v2 U3(
    .vidon(vidon),
    .hc(hc),
    .vc(vc),

    .red(red),
    .green(green),
    .blue(blue),

    .btn(btn),
    .move(move),
    .rotate(rotate),
    .clk(clk_out)
    );
```

in VGA_top.v

```
`include "triangle.v"
`include "triangle_x.v"
`include "square.v"
`include "parallelogram.v"

module VGA_display_v2(input wire vidon,
                      input wire [10:0] hc,
                      vc,
                      (* dont_touch = "true" *) input wire [7:0] btn,
                      (* dont_touch = "true" *) input wire [3:0] move,
                      input wire rotate,
                      input wire clk,
                      output reg [3:0] red,
```

```
                    green,
                    blue);

    wire [7:0] color;

//     the middle one, which right edge are both along axis (h or v)
    triangle U_100(
        .vidon(vidon),
        .hc(hc),
        .vc(vc),
        .btn(btn[4]),
        .move(move),
        .rotate(rotate),
        .color(color[4]),
        .clk(clk)
    );

    // the follows 4 triangle's long edge all are along axis
    // small
   triangle_x U_110(
        .vidon(vidon),
        .hc(hc),
        .vc(vc),
        .sb(0),
        .btn(btn[6]),
        .move(move),
        .rotate(rotate),
        .color(color[6]),
        .clk(clk)
   );

   // small
   triangle_x U_001(
        .vidon(vidon),
        .hc(hc),
        .vc(vc),
        .sb(0),
        .btn(btn[1]),
        .move(move),
        .rotate(rotate),
        .color(color[1]),
        .clk(clk)
   );

   // big
   triangle_x U_010(
        .vidon(vidon),
        .hc(hc),
        .vc(vc),
        .sb(1),
        .btn(btn[2]),
        .move(move),
        .rotate(rotate),
        .color(color[2]),
        .clk(clk)
   );

   // big
```

```
    triangle_x U_111(
         .vidon(vidon),
         .hc(hc),
         .vc(vc),
         .sb(1),
         .btn(btn[7]),
         .move(move),
         .rotate(rotate),
         .color(color[7]),
         .clk(clk)
    );

    square U_101(
         .vidon(vidon),
         .hc(hc),
         .vc(vc),
         .btn(btn[5]),
         .move(move),
         .color(color[5]),
         .clk(clk)
    );

    parallelogram U_011(
         .vidon(vidon),
         .hc(hc),
         .vc(vc),
         .btn(btn[3]),
         .move(move),
         .rotate(rotate),
         .color(color[3]),
         .clk(clk)
    );

     // sum up all the color
     always @(*) begin
         red   = 0;
         green = 0;
         blue  = 0;

         if(vidon) begin
            if (color[4] == 1 || color[5] == 1 || color[6] == 1 || color[7] == 1)
                red = 4'b1111;
            if (color[2] == 1 || color[3] == 1 || color[6] == 1 || color[7] == 1)
                green = 4'b1111;
            if (color[1] == 1 || color[3] == 1 || color[5] == 1 || color[7] == 1)
                blue = 4'b1111;
         end
     end

endmodule //VGA_display_v2
```

VGA_display_v2.v

And here is a sample of its sub-module:

```
// The right edge is not horizontal
// The centre point(h0, v0) is the mid-point of long edge
```

```
module triangle_x(
    input wire vidon,
    input wire [10:0] hc,
    vc,
    input wire sb, // 0 present small one (right edge = 100r2), 1 present big
one (200r2).
    input wire btn,
    input wire [3:0] move,
    input wire rotate,
    input wire clk,
    output reg color
);

    // @Parameter
    integer  h_min   = 215; // original of h
    integer  v_min   = 26;  // original of v
    integer  h_max   = 1015;
    integer  v_max   = 626;
    integer  size_1  = 100;

    integer h0, v0;
    integer hc_s, vc_s, btn_s, rotate_s;

    integer cnt_move;
    integer cnt_rotate;
    integer one_s;

    integer realSize;

    always@(*) begin
        hc_s = $unsigned(hc);
        vc_s = $unsigned(vc);
        btn_s = $unsigned(btn);
        rotate_s = $unsigned(rotate);
    end

    // Initialization
    initial begin
        h0         = h_min + 400;
        v0         = v_min + 300;
        cnt_move   = 0;
        cnt_rotate = 0;
        one_s = 0;
    end

    always @(negedge clk) begin
        if (sb == 0) realSize = 100;
        else realSize = 200;

        // Move along h or v
        if (hc_s == h_min && btn_s == 1) begin
            if ((move[0] == 1 || move[1] == 1 || move[2] == 1 || move[3] == 1)
&& cnt_move <= 506)
                cnt_move <= cnt_move + 1;
            else if (move[0] == 1 || move[1] == 1 || move[2] == 1 || move[3] ==
1) begin
                cnt_move <= 0;
```

```
                if (move[0] == 1 && v0 > v_min + size_1)        v0 <= v0 - 1;
                else if (move[1] == 1 && v0 < v_max - size_1)   v0 <= v0 + 1;
                else if (move[2] == 1 && h0 > h_min + size_1)   h0 <= h0 - 1;
                else if (move[3] == 1 && h0 < h_max - size_1)   h0 <= h0 + 1;
            end
        end

        // if rotate in hign e-level, then cnt_rotate += 1;
        if (hc_s == h_min && vc_s == v_min) begin
            if (one_s == 60) begin
                one_s <= 0;
                if (btn_s == 1 && rotate_s == 1) begin
                    if (cnt_rotate == 3) cnt_rotate <= 0;
                    else cnt_rotate <= cnt_rotate + 1;
                end
            end
            else one_s <= one_s + 1;
        end

        // Display of triangle
        if (vidon == 1) begin
            case (cnt_rotate)
                2'b00:  if (vc_s < v0 && (hc_s + vc_s) > (h0 + v0) - realSize
                            && (hc_s - vc_s) < (h0 - v0) + realSize) color <= 1;
                        else color <= 0;
                2'b01:  if (hc_s > h0 && (hc_s - vc_s) < (h0 - v0) + realSize
                            && (hc_s + vc_s) < (h0 + v0) + realSize) color <= 1;
                        else color <= 0;
                2'b10:  if (vc_s > v0 && (hc_s + vc_s) < (h0 + v0) + realSize
                            && (hc_s - vc_s) > (h0 - v0) - realSize) color <= 1;
                        else color <= 0;
                2'b11:  if (hc_s < h0 && (hc_s - vc_s) > (h0 - v0) - realSize
                            && (hc_s + vc_s) > (h0 + v0) - realSize) color <= 1;
                        else color <= 0;
                default: color <= 0;
            endcase
        end
        else color <= 0;
    end

endmodule // triangle_x
```

triangle_x.v

## Module U4 & U5

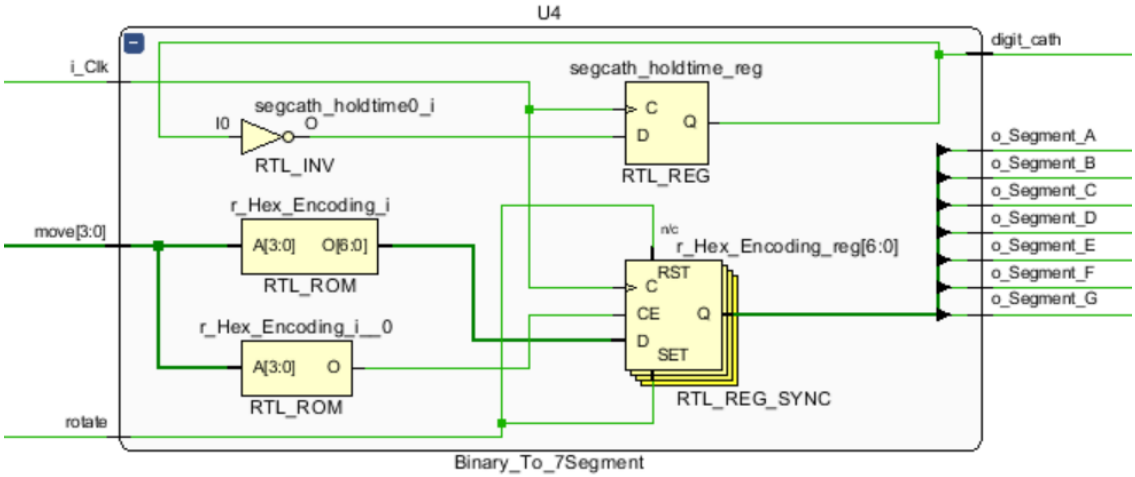
(Binary_To_7Segment) & (Binary_To_7Segments)

- **Function**: U5 control the leftmost 7-segment light to display {0, 1, 2, 3, 4, 5, 6, 7};

  U4 control the rightmost 7-segment light to display {U, D, L, R, S}.
- **I/O**: input: "i_Clk", "move" and "rotate" (U5 use "btn");

  output: "digit_cath" (determine the light position), "o_Segment_A/B/C/D/E/F/G" (7-Segment).
- **RTL**:

U4 as sample:




- **Code**:

  U4 as sample:

```
module Binary_To_7Segment
  (
   input        i_Clk,
   input wire [3:0] move,
   input wire rotate,
   output       o_Segment_A,
   output       o_Segment_B,
   output       o_Segment_C,
   output       o_Segment_D,
   output       o_Segment_E,
   output       o_Segment_F,
   output       o_Segment_G,
   output [0:0] digit_cath
   );

  reg [6:0]     r_Hex_Encoding = 7'h00;

  // Purpose: Creates a case statement for all possible input binary
numbers.
  // Drives r_Hex_Encoding appropriately for each input combination.
  always @(posedge i_Clk)
    begin
      case (move)
        4'b0001 : r_Hex_Encoding <= 7'h3e;//up
        4'b0010 : r_Hex_Encoding <= 7'h5e;//down
        4'b0100 : r_Hex_Encoding <= 7'h38;//left
        4'b1000 : r_Hex_Encoding <= 7'h31;//right
        //5'b10000 : r_Hex_Encoding <= 7'h49;//spin
      endcase
      case(rotate)
      1'b1:r_Hex_Encoding<=7'h49;
      endcase
    end
    reg segcath_holdtime;
    always @ (posedge i_Clk)

    begin
```

```
    segcath_holdtime<=~segcath_holdtime;
    end
    assign digit_cath={segcath_holdtime};

  // r_Hex_Encoding[7] is unused
  assign o_Segment_A = r_Hex_Encoding[0];
  assign o_Segment_B = r_Hex_Encoding[1];
  assign o_Segment_C = r_Hex_Encoding[2];
  assign o_Segment_D = r_Hex_Encoding[3];
  assign o_Segment_E = r_Hex_Encoding[4];
  assign o_Segment_F = r_Hex_Encoding[5];
  assign o_Segment_G = r_Hex_Encoding[6];

endmodule // Button_To_7Segment
```

Binary_To_7Segment.v

# 3 Testbench

***Because the purpose of the project is to display in the screen and the geometric figure changing is hard to find in the testbench, I only write the testbench of top file (VGA_top_tb.v). All of its sub-module could be find in this testbench : )**

- **Test purpose**:

  (1) All the value is not in z/Z or X (High resistance state and indeterminate state);

  (2) "hsync" and "vsync" (行同步和场同步) are harmonious;

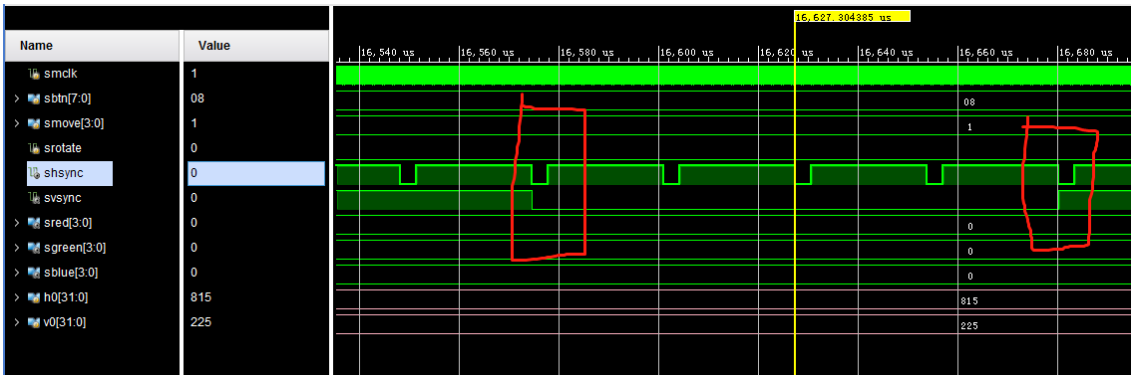
  (3) Color output is normal;

  (4) Move and Rotation is normal

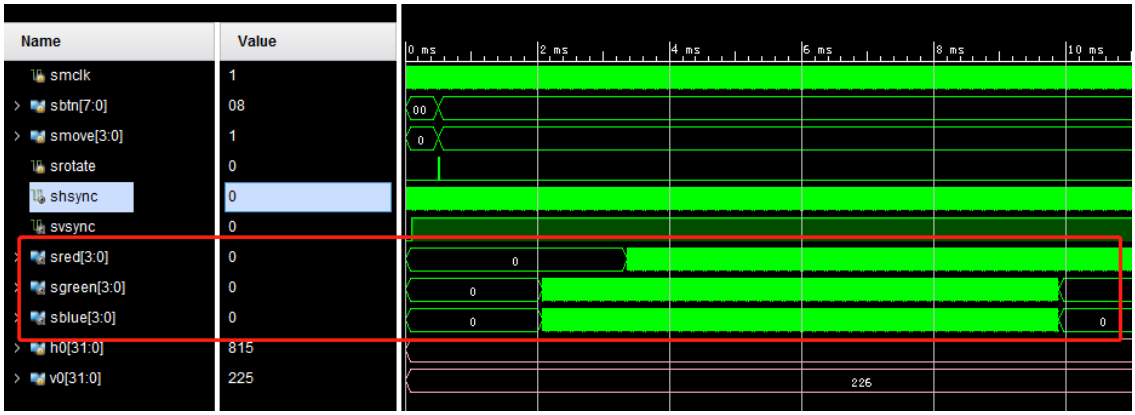
- **Result**:

  (1) You could see (2) (3) (4) has no z/Z or X

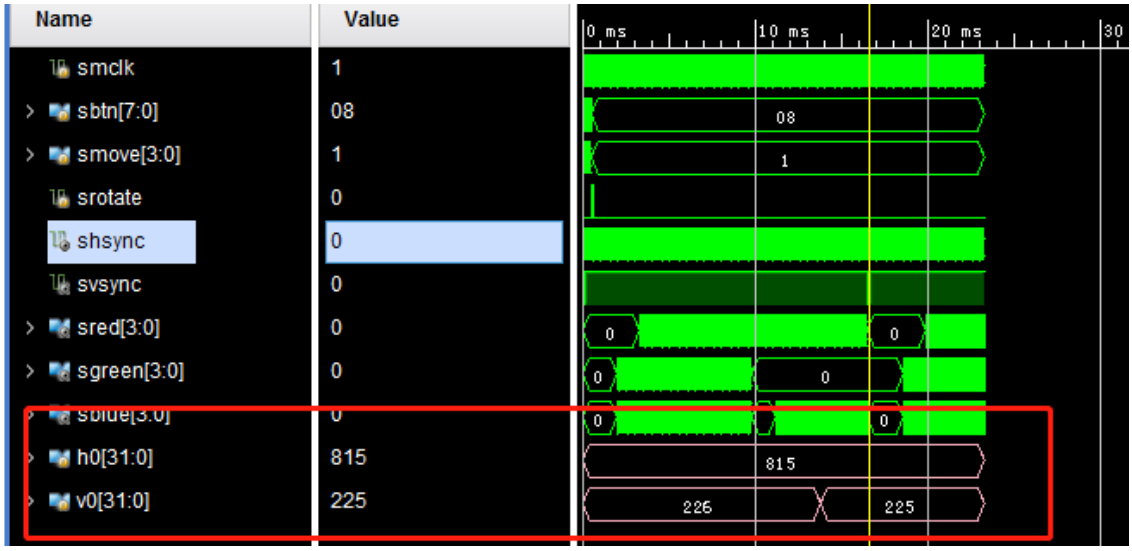
  (2)




  (3)




(4)




- **Code**:

```
`timescale 1ns / 1ps
`include "VGA_top.v"

module VGA_top_tb();
    reg smclk;
    reg [7:0] sbtn;
    reg [3:0] smove;
    reg srotate;
    wire shsync, svsync;
    wire [3:0] sred, sgreen, sblue;

    VGA_top U(smclk, sbtn, smove, srotate, shsync, svsync, sred, sgreen,
sblue);

    initial begin
        $dumpfile("VGA_top_tb.vcd");
        $dumpvars;
        smclk  = 0;
        sbtn   = 0;
        smove  = 0;
        srotate = 0;
        repeat(100000) # 5 smclk = ~smclk;
        # 5 sbtn = 16;
        # 5 smove = 1;
        # 5 srotate = 1;
        # 5000 srotate = 0;
        repeat(100000000) # 5 smclk = ~smclk;
```

```
        repeat(100000) # 5 smclk = ~smclk;


        # 10 $finish;
    end
endmodule // VGA_top_tb
```

# 4 Result / screen-shot

As the figures show:

stripes

Some geometric figures

before operations

after

# 5 Summary & feeling-and-experience

*(self-evaluate is also in this part)*

**Summary**:

In this project, our team successfully completed all the basic parts and explored some advanced parts. VGA was a bit more difficult to begin than the other problems, and we had worked on the first 20 points, rainbow stripes, for two weeks. Because VGA could not be connected to the display screen, the resolution was not suitable, and the timing design, we were stuck for a while. In the subsequent drawing content, we were also stuck due to Bresenham algorithm and other problems. Later we found that it's much easier to draw a fill rather than a straight line. As we became more familiar with hardware programming, the translation and rotation became easier to solve. But the input of pins, the wrong use of combined circuits, merging modules and other problems, still let us waste much time. Finally, our group's project came to a perfect end.

**YU Kunyi**:

In the entire programme project, I really learn a lot and basically master the Verilog. Although I am often struggling in debug, when we find the result is great, it's really please me.

I think I take most of the workload of code and debug that's why I choose 60% workload. But my teammate is also very responsible and make PPT and video which also save my time.

Wish in next semester 计组 lessons, I will be better treat by Verilog ：) Haaaaaaa!

# 6 References


[1] 康磊. (2010). 数字电路设计及Verilog HDL实现. 西安: 西安电子科技大学出版社.

[2] 韦克利, 林生, 葛红, & 金京林. (2019). 数字设计 原理与实践 (计算机科学丛书 Ji Suan Ji Ke Xue Cong Shu). 北京: 机械工业出版社.

[3] 汉纳, 郑利浩, 王荃, & 陈华锋. (2010). FPGA数字逻辑设计教程 Verilog. 北京: 电子工业出版社.

[4] Nathan Ickes. (April 29, 2004). *VGA Video. Boston, Massachusetts: MIT 6.111 Introduction to Digital Systems. Retrieved from* https://web.mit.edu/6.111/www/s2004/NEWKIT/vga.shtml


**We also upload a video on bilibili: yeah!**

【数字逻辑（CS207）春季期末project-VGA七巧板-哔哩哔哩】 https://b23.tv/YQsKlWa