

# verilog FPGA VGA 显示

显示器扫描方式分为逐行扫描和隔行扫描：逐行扫描是扫描从屏幕左上角一点开始，从左像右逐点扫描，每扫描完一行，电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，开始下一帧。隔行扫描是指电子束扫描时每隔一行扫一线，完成一屏后在返回来扫描剩下的线，**隔行扫描的显示器闪烁的厉害，会让使用者的眼睛疲劳。**

完成一行扫描的时间称为水平扫描时间，其倒数称为行频率；完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率，即刷新一屏的频率，常见的有 60Hz，75Hz 等等。标准的 VGA 显示的场频 60Hz，行频 31.5KHz。

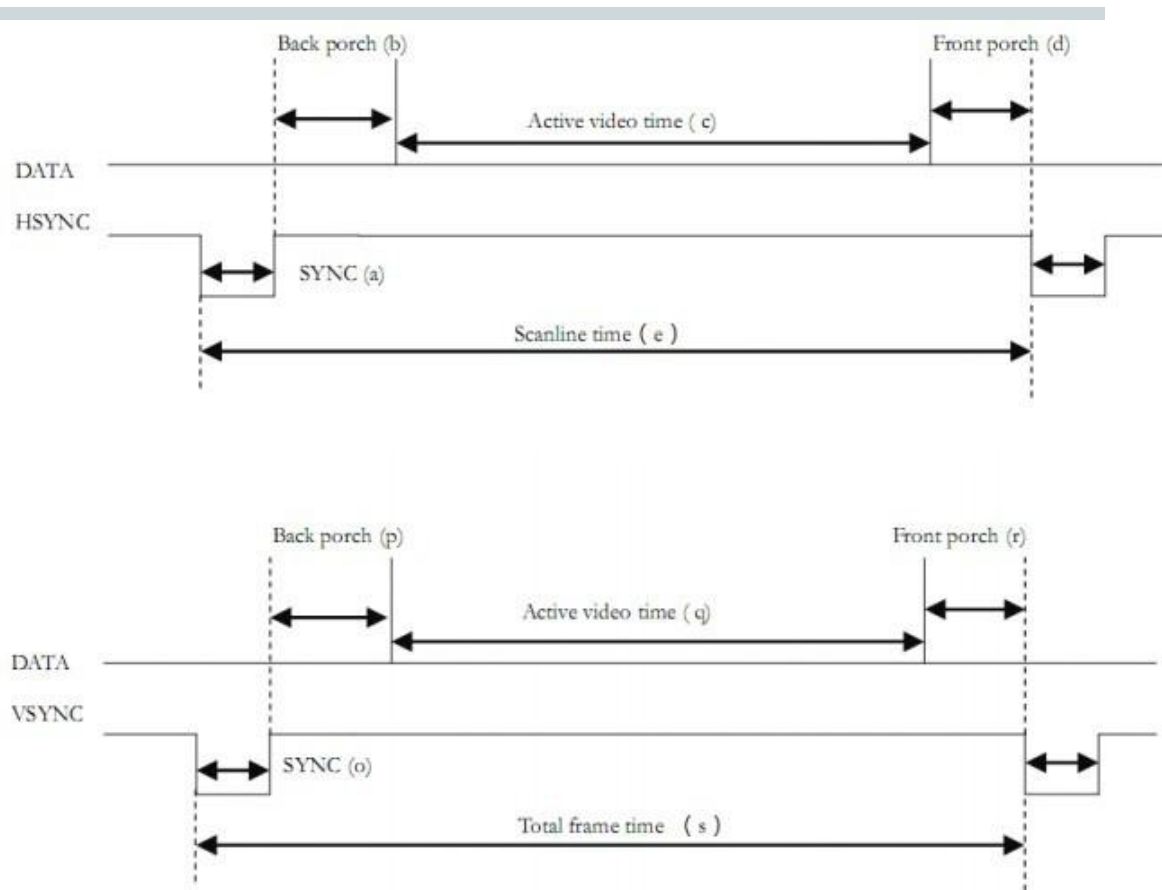
行场消隐信号：是针对老式显像管的成像扫描电路而言的。电子枪所发出的电子束从屏幕的左上角开始向右扫描，一行扫完需将电子束从右边移回到左边以便扫描第二行。在移动期间就必须有一个信号加到电路，使得电子束不能发出。不然这个回扫线会破坏屏幕图像的。这个阻止回扫线产生的信号就叫作消隐信号，场信号的消隐也是一个道理。

显示带宽：带宽指的显示器可以处理的频率范围。如果是 60Hz 刷新频率的 VGA，其带宽达  $640 \times 480 \times 60 = 18.4\text{MHz}$ ，70Hz 的刷新频率 1024x768 分辨率的 SVGA，其带宽达  $1024 \times 768 \times 70 = 55.1\text{MHz}$ 。

时钟频率：以 640x480@59.94Hz (60Hz) 为例，每场对应 525 个行周期 ( $525 = 10 + 2 + 480 + 33$ )，其中 480 为显示行。每场有场同步信号，该脉冲宽度为 2 个行周期的负脉冲，每显示行包括 800 点时钟，其中 640 点为有效显示区，每一行有一个行同步信号，该脉冲宽度为 96 个点时钟。由此可知：行频为  $525 \times 59.94 = 31469\text{Hz}$ ，**需要点时钟频率：525\*800\*59.94 约 25MHz。**

一、VGA 时序分析：





VESA 中定义行时序和场时序都需要同步脉冲 (Sync a)、显示后沿 (Back porch b)、显示时序段 (Display interval c) 和显示前沿 (Front porch d) 四部分。VGA 工业标准显示模式要求：**行同步，场同步都为负极性，即同步脉冲要求是负脉冲。**

由 VGA 的行时序可知：没一行都有一个负极性行同步脉冲 (Sync a)，是数据行的结束标志，同时也是下一行的开始标志。在同步脉冲之后为显示后沿 (Back porch b)，在显示时序段 (Display interval c) 显示器为亮的过程，RGB 数据驱动一行上的每一个像素点，从而显示一行。在一行的最后为显示前沿 (Front porch d)。在显示时间段 (Display interval c) 之外没有图像投射到屏幕是插入消隐信号。同步脉冲 (Sync a)、显示后沿 (Back porch b) 和显示前沿 (Front porch d) 都是在行消隐间隔内 (Horizontal Blanking Interval)，当消隐有效时，RGB 信号无效，屏幕不显示数据。

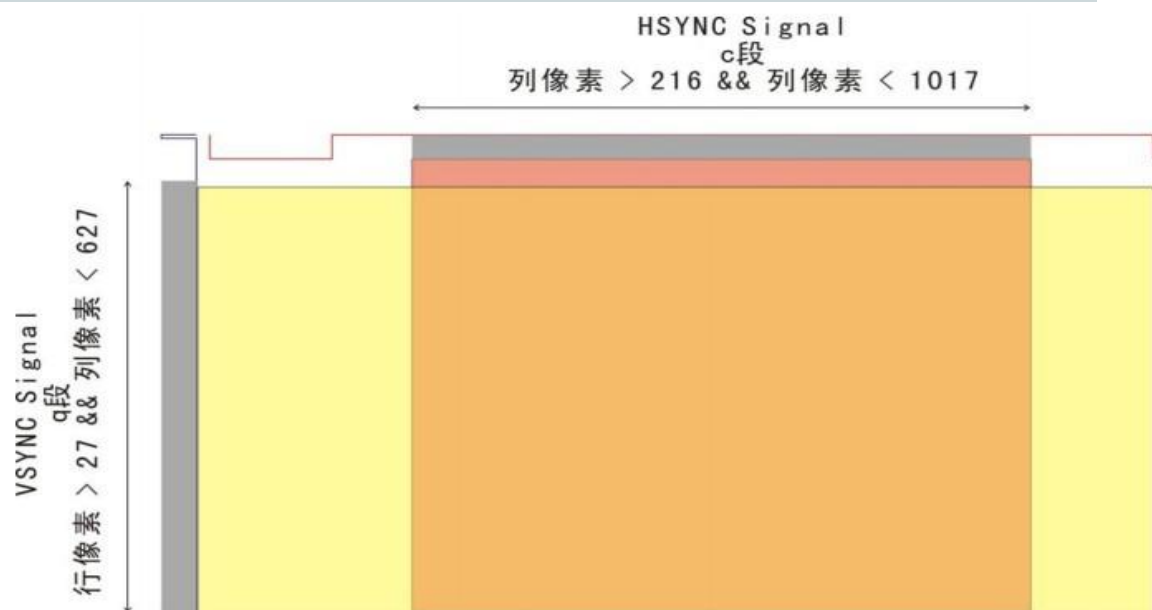
VGA 的场时序与行时序基本一样，每一帧的负极性脉冲 (Sync a) 是一帧的结束标志，同时也是下一帧的开始标志。而显示数据是一帧的所有行数据。

下面以 800x600@60Hz 分辨率威力详细讲解 VGA 时序：

800 x 600 x 60Hz	a 段	b 段	c 段	d 段	e 段-总共 n 个列像素
HSYNC Signal 列像素	128	88	800	40	1056
800 x 600 x 60Hz	o 段	p 段	q 段	r 段	s 段-总共 n 个行像素
VSYNC Signal 行像素	4	23	600	1	628

HSYNC Signal 是用来控制“列填充”，而一个 HSYNC Signal 可以分为 4 个段，也就是 a(同步段)，b(后廊段)，c(激活段)，d(前廊段)。HSYNC Signal 的 a 是拉低的 128 个列像素，b 是拉高的 88 个列像素，至于 c 是拉高的 800 个列像素，而最后的 d 是拉高的 40 个列像素。一行总共有 1056 个列像素。

VSYNC Signal 是用来控制“行扫描”。而一个 VSYNC Signal 同样可以分为 4 个段，也是 o(同步段)，p(后廊段)，q(激活段)，r(前廊段)。VSYNC Signal 的 o 是拉低的 4 个行像素，p 是拉高的 23 个行像素，至于 q 是拉高的 600 个行像素，而最后的 r 是拉高的 1 个行像素。一行总共有 628 个行像素。



在上图表示了，HSYNC Signal 只有有的 C 段（红色部分）和 VSYNC Signal 的 q 段（黄色部分）的激活段，数据的输入才有效。换句话说，显示图片是发生在交叉（橘色部分）的“有效区域”下。

交叉部分的表达式可以如此描述：

列像素 > 216 && 列像素 < 1017 && 行像素 > 27 && 行像素 < 627。

VGA 800x600@60Hz 所需时钟频率:  $VGA\_CLK=1056 \times 628 \times 60=39790080 \sim 40\text{MHz}$ ;

CLK //50MHZ 时钟

CTL//控制端, 控制彩条的四种方式

RED// 红色, 经电阻分出三个端口, 构成 8 级红

GREEN//绿色, 经电阻分出三个端口, 构成 8 级绿

BLUE//蓝色, 经电阻分出 2 个端口, 构成 4 级红

HS//行同步信号

VS//场同步信号

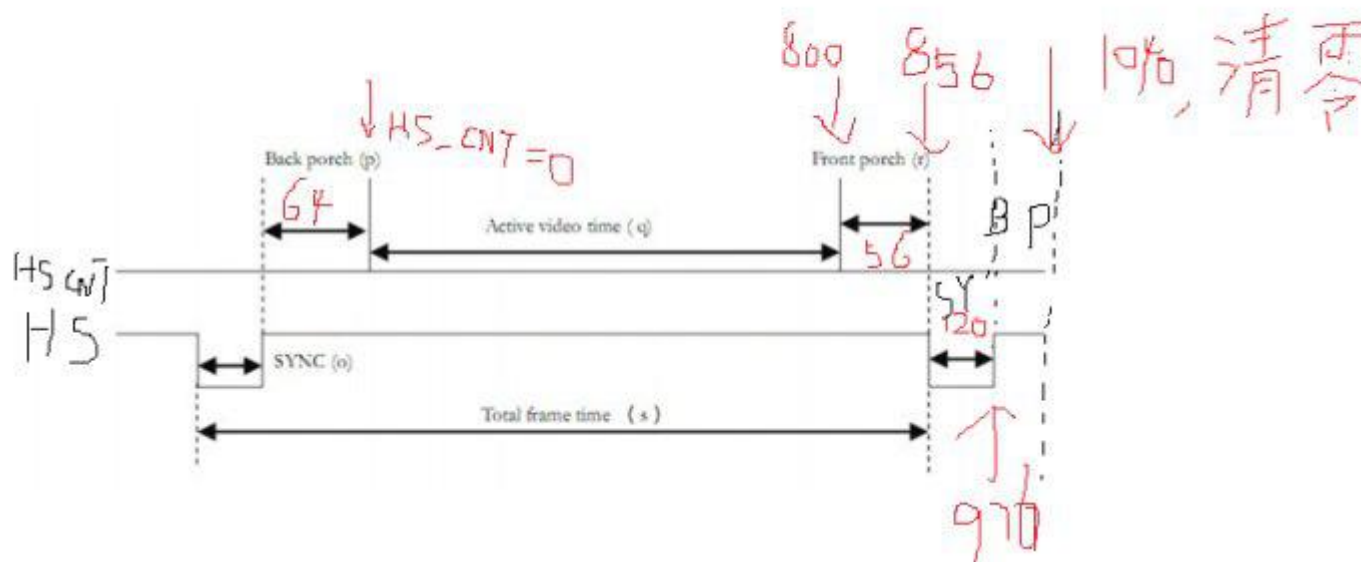
800X600@72HZ,

pixel clock: 50MHZ. 行 ACTIVE PIXELS 800, 行 FP pixels56, sync pixels 120, 行 BP pixels 64, 总共的 pixels1040

场 active pixels 600, 场 FP PIXELS37, SYNC pixels 6, 场 BP 23 ,总共的 pixels 666

程序主体来自 CSDN 博客, 转载请标明出处:

<http://blog.csdn.net/chevroletss/archive/2010/01/15/5195987.aspx>



```

1. //VGA DISPLAY
2. //////////////////////////////////////
3.
4. module VGA_DISPLAY(
5. CLK,
6. CTL,
7. RED,
8. GREEN,
9. BLUE,
10. HS,
11. VS);
12.     input CLK;
13.     input [1:0] CTL;
14.     output [2:0] RED, GREEN;
15.     output [1:0] BLUE;
16.     output HS, VS;
17.
18.     reg HS, VS;
19.     reg [11:0] HS_CNT;// 行计数
20.     reg [9:0] VS_CNT;//场计数
21.     reg [2:0] RED, GREEN;
22.     reg [1:0] BLUE;
23.     reg [2:0] RED_H=3'b000, GREEN_H=3'b000;
24.     reg [1:0] BLUE_H=2'b00;
25.     reg [2:0] RED_V=3'b000, GREEN_V=3'b000;
26.     reg [1:0] BLUE_V=2'b00;
27.
28.
29.     always @(posedge CLK)
30.     begin
31.         if(857<=HS_CNT&&HS_CNT<=977) HS<=0;//产生 HS 信号
32.         else HS<=1;
33.         if(HS_CNT==1039) begin
34.             HS_CNT<=0;
35.             if(VS_CNT==665) VS_CNT<=0;//VS 计
数
36.             else VS_CNT<=VS_CNT+1; end
37.         else HS_CNT<=HS_CNT+1;//HS 计数
38.         if(638<=VS_CNT&&VS_CNT<=644) VS<=0;//产生 VS 信号
39.         else VS<=1;
40.     end
41.
42.

```

```

43.     always @(posedge CLK)
44.     begin
45.         if ((0<=HS_CNT&&HS_CNT<=266)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b0000_0011;
46.         else if ((267<=HS_CNT&&HS_CNT<=534)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b1110_0000;
47.         else if ((534<=HS_CNT&&HS_CNT<=799)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b0001_1100;
48.         else {RED_H, GREEN_H, BLUE_H} <=8'b0000_0000;
49.     end
50.     always @(posedge CLK)
51.     begin
52.         if ((0<=HS_CNT&&HS_CNT<=799)&&(0<=VS_CNT&&VS_CNT<=199)) {RED_V, GREEN_V, BLUE_V} <=8'b0000_0011;
53.         else if ((0<=HS_CNT&&HS_CNT<=799)&&(200<=VS_CNT&&VS_CNT<=399)) {RED_V, GREEN_V, BLUE_V} <=8'b1110_0000;
54.         else if ((0<=HS_CNT&&HS_CNT<=799)&&(400<=VS_CNT&&VS_CNT<=599)) {RED_V, GREEN_V, BLUE_V} <=8'b0001_1100;
55.         else {RED_V, GREEN_V, BLUE_V} <=8'b0000_0000;
56.     end
57.
58.
59.     always @(CTL or RED_H or GREEN_H or BLUE_H or RED_V or GREEN_V or BLUE_V)
60.     begin
61.         case(CTL) //控制显示模式
62.             2'b00: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H} ; end //竖条
63.             2'b01: begin {RED, GREEN, BLUE} = {RED_V, GREEN_V, BLUE_V} ; end //横条
64.             2'b10: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H}+{RED_V, GREEN_V, BLUE_V} ; end //方块
65.             2'b11: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H}-{RED_V, GREEN_V, BLUE_V} ; end
66.         endcase
67.     end
68.
69. endmodule

```

下面的程序 分别显示了一个长方形的蓝色小块和一个红色的（非矩形）小球位图

```

module VGA_DISPLAY
(CLK,
SW,

```

```

RED,
GREEN,
BLUE,
HS,
VS,alarm);
    input CLK;
    input [1:0] SW;
    output [2:0] RED, GREEN;
    output [1:0] BLUE;
    output HS, VS;
    output alarm;
    reg HS, VS;
    reg [11:0] HS_CNT;// 行计数
    reg [9:0] VS_CNT;//垂直计数
    reg [2:0] RED, GREEN;
    reg [1:0] BLUE;
    reg [2:0] RED_H=3' b000, GREEN_H=3' b000;
    reg [1:0] BLUE_H=2' b00;
    reg [2:0] RED_V=3' b000, GREEN_V=3' b000;
    reg [1:0] BLUE_V=2' b00;
    wire alarm;
    assign alarm=1' b0;
    always @(posedge CLK)
    begin
        if (857<=HS_CNT&&HS_CNT<=965) HS<=0;//产生 HS 信号
        else HS<=1;
        if (HS_CNT==1039) begin
            HS_CNT<=0;
        if (VS_CNT==660) VS_CNT<=0;//VS 计数
            else VS_CNT<=VS_CNT+1;
        end
        else HS_CNT<=HS_CNT+1;//HS 计数
        if (638<=VS_CNT&&VS_CNT<=644) VS<=0;//产生 VS 信号
        else VS<=1;
    end

wire [2:0] rom_addr,room_col;
reg[7:0] rom_data;

always @*
case(rom_addr)
    3'h0:rom_data=8' b00111100;
    3'h1:rom_data=8' b01111110;

```

```

3'h2:rom_data=8'b11111111;
3'h3:rom_data=8'b11111111;
3'h4:rom_data=8'b11111111;
3'h5:rom_data=8'b11111111;
3'h6:rom_data=8'b01111110;
3'h7:rom_data=8'b00111100;
endcase

assign ball_area=((100<=HS_CNT)&&(HS_CNT<=108)&&(200<=VS_CNT)&&(VS_CNT<=208));
assign rom_addr = VS_CNT - 200;
assign rom_col = HS_CNT - 100;
assign rom_bit = rom_data[rom_col];
assign ball_on = ball_area&&rom_bit;

always @(posedge CLK)
begin
    if((400<=HS_CNT&&HS_CNT<=500)&&(80<=VS_CNT&&VS_CNT<=100))    {RED_V, GREEN_V, BLUE_V} <=8'b0000_0011;

    else if (ball_on)

{RED_V, GREEN_V, BLUE_V} <=8'b1110_0000;

    else    begin
        {RED_V, GREEN_V, BLUE_V} <=8'b0000_0000;
    end
end

always @*
begin

{RED, GREEN, BLUE}    = {RED_V, GREEN_V, BLUE_V} ;

end

endmodule

```

以下是按键改变 VGA 颜色的视频

[http://v.youku.com/v\\_show/id\\_XNDY30TQxMTIw.html](http://v.youku.com/v_show/id_XNDY30TQxMTIw.html)



CLK //50MHZ 时钟

CTL//控制端，控制彩条的四种方式

RED// 红色，经电阻分出三个端口，构成 8 级红

GREEN//绿色，经电阻分出三个端口，构成 8 级绿

BLUE//蓝色，经电阻分出 2 个端口，构成 4 级红

HS//行同步信号

VS//场同步信号

800X600@72HZ,

pixel clock: 50MHZ. 行 ACTIVE PIXELS 800, 行 FP pixels56, PW pixels 120, 行 BP pixels 64, 总共的 pixels1040

场 active pixels 600, 场 FP PIXELS37, PW pixels 6, 场 BP 23 , 总共的 pixels 666

结果很完美

[\[c-sharp\]](#) [view plaincopy](#)

```
1.  ////////////////////////////////////////////
2.  //VGA DISPLAY
3.  ////////////////////////////////////////////
4.
5.  module VGA_DISPLAY(
6.  CLK,
7.  CTL,
8.  RED,
9.  GREEN,
10. BLUE,
11. HS,
12. VS);
13.  input CLK;
14.  input [1:0] CTL;
15.  output [2:0] RED, GREEN;
16.  output [1:0] BLUE;
17.  output HS, VS;
18.
19.  reg HS, VS;
20.  reg [11:0] HS_CNT; // 行计数
21.  reg [9:0] VS_CNT; // 场计数
22.  reg [2:0] RED, GREEN;
23.  reg [1:0] BLUE;
```

```

24. reg [2:0] RED_H=3'b000, GREEN_H=3'b000;
25. reg [1:0] BLUE_H=2'b00;
26. reg [2:0] RED_V=3'b000, GREEN_V=3'b000;
27. reg [1:0] BLUE_V=2'b00;
28.
29.
30. always @(posedge CLK)
31. begin
32.     if(857<=HS_CNT&&HS_CNT<=977) HS<=0; //产生 HS 信号
33.     else HS<=1;
34.     if(HS_CNT==1039) begin
35.         HS_CNT<=0;
36.         if(VS_CNT==665) VS_CNT<=0; //VS 计数
37.         else VS_CNT<=VS_CNT+1; end
38.     else HS_CNT<=HS_CNT+1; //HS 计数
39.     if(638<=VS_CNT&&VS_CNT<=644) VS<=0; //产生 VS 信号
40.     else VS<=1;
41. end
42.
43.
44. always @(posedge CLK)
45. begin
46.     if((0<=HS_CNT&&HS_CNT<=266)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b0000_0011;
47.     else if((267<=HS_CNT&&HS_CNT<=534)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b1110_0000;
48.     else if((534<=HS_CNT&&HS_CNT<=799)&&(0<=VS_CNT&&VS_CNT<=599)) {RED_H, GREEN_H, BLUE_H} <=8'b0001_1100;
49.     else {RED_H, GREEN_H, BLUE_H} <=8'b0000_0000;
50. end
51. always @(posedge CLK)
52. begin
53.     if((0<=HS_CNT&&HS_CNT<=799)&&(0<=VS_CNT&&VS_CNT<=199)) {RED_V, GREEN_V, BLUE_V} <=8'b0000_0011;
54.     else if((0<=HS_CNT&&HS_CNT<=799)&&(200<=VS_CNT&&VS_CNT<=399)) {RED_V, GREEN_V, BLUE_V} <=8'b1110_0000;
55.     else if((0<=HS_CNT&&HS_CNT<=799)&&(400<=VS_CNT&&VS_CNT<=599)) {RED_V, GREEN_V, BLUE_V} <=8'b0001_1100;
56.     else {RED_V, GREEN_V, BLUE_V} <=8'b0000_0000;
57. end
58.
59.
60. always @(CTL or RED_H or GREEN_H or BLUE_H or RED_V or GREEN_V or BLUE_V)

```

```

61. begin
62.     case(CTL) //控制显示模式
63.         2'b00: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H} ; end //竖条
64.         2'b01: begin {RED, GREEN, BLUE} = {RED_V, GREEN_V, BLUE_V} ; end //横条
65.         2'b10: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H} + {RED_V, GREEN_V, BLUE_V} ; end //方块
66.         2'b11: begin {RED, GREEN, BLUE} = {RED_H, GREEN_H, BLUE_H} - {RED_V, GREEN_V, BLUE_V} ; end
67.     endcase
68. end
69.
70. endmodule
71.

```

## 端口定义

[\[c-sharp\]](#) [view plain](#) [copy](#)

```

1. NET "CLK" LOC = "B8";
2. NET "CTL<0>" LOC = "G18";
3. NET "CTL<1>" LOC = "H18";
4. NET "RED<0>" LOC = "R9";
5. NET "RED<1>" LOC = "T8";
6. NET "RED<2>" LOC = "R8";
7. NET "GREEN<0>" LOC = "N8";
8. NET "GREEN<1>" LOC = "P8";
9. NET "GREEN<2>" LOC = "P6";
10. NET "BLUE<0>" LOC = "U5";
11. NET "BLUE<1>" LOC = "U4";
12.
13. NET "HS" LOC = "T4";
14. NET "VS" LOC = "U3";

```