同济大学计算机系

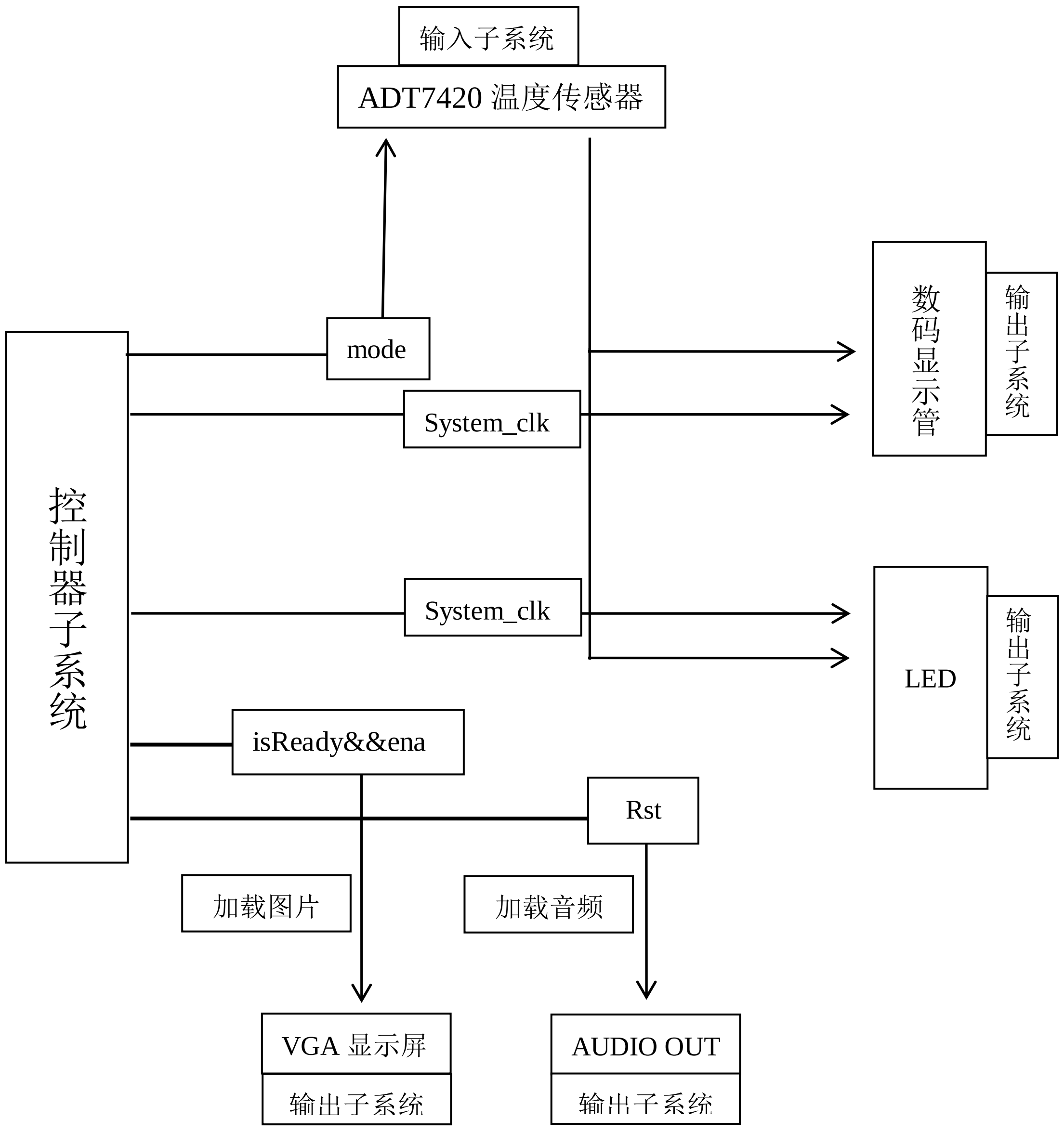
数字逻辑课程实验报告

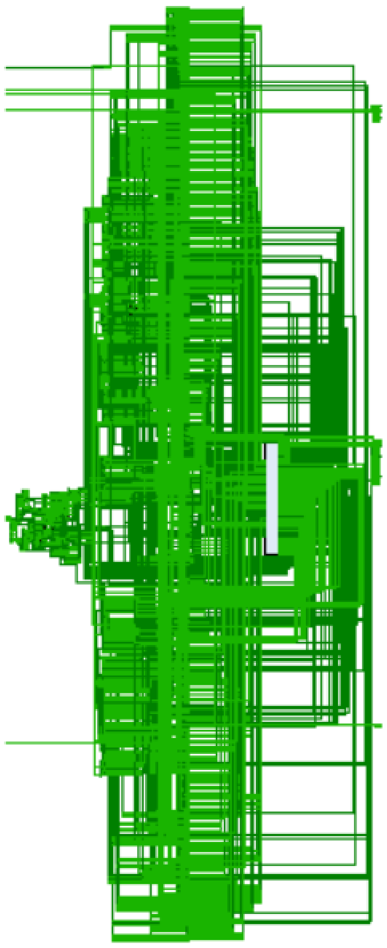
学　　号	1652286
姓　　名	李辉
专　　业	计算机
授课老师	张冬冬

一、实验内容

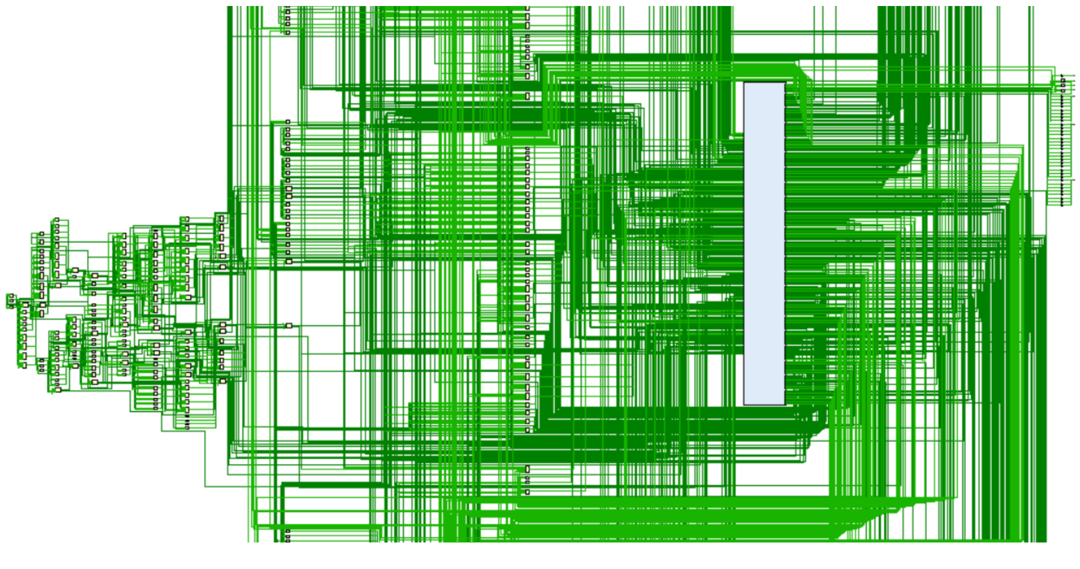
基于 ADT7420 温度传感器，VGA 显示屏，七段数码显示管及 LED 灯，板置音频输出设计实现的具有温度显示，音乐播放的**多功能系统**。

二、多功能系统总框图



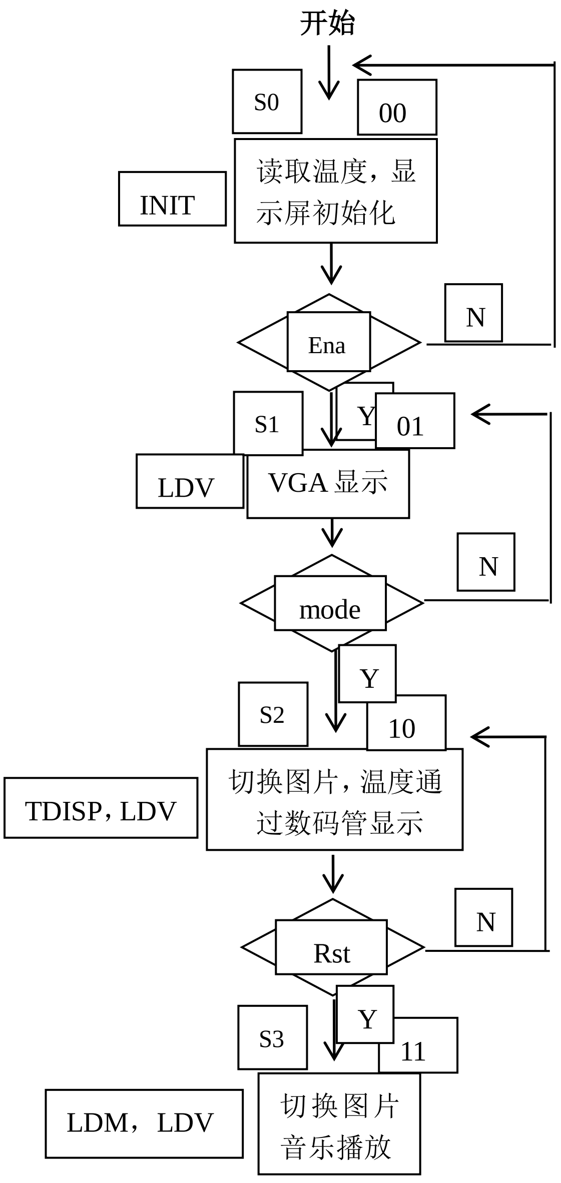


Schematic 电路图



三、系统控制器设计

1、ASM 流程图



2、状态转移真值表

PS			NS			转换条件
状态	B	A	状态	B(D)	A(D)	
S0	0	0	S1	0	1	Ena
S1	0	1	S2	1	0	mode
S2	1	0	S3	1	1	Rst
S3	1	1				

次态激励函数：

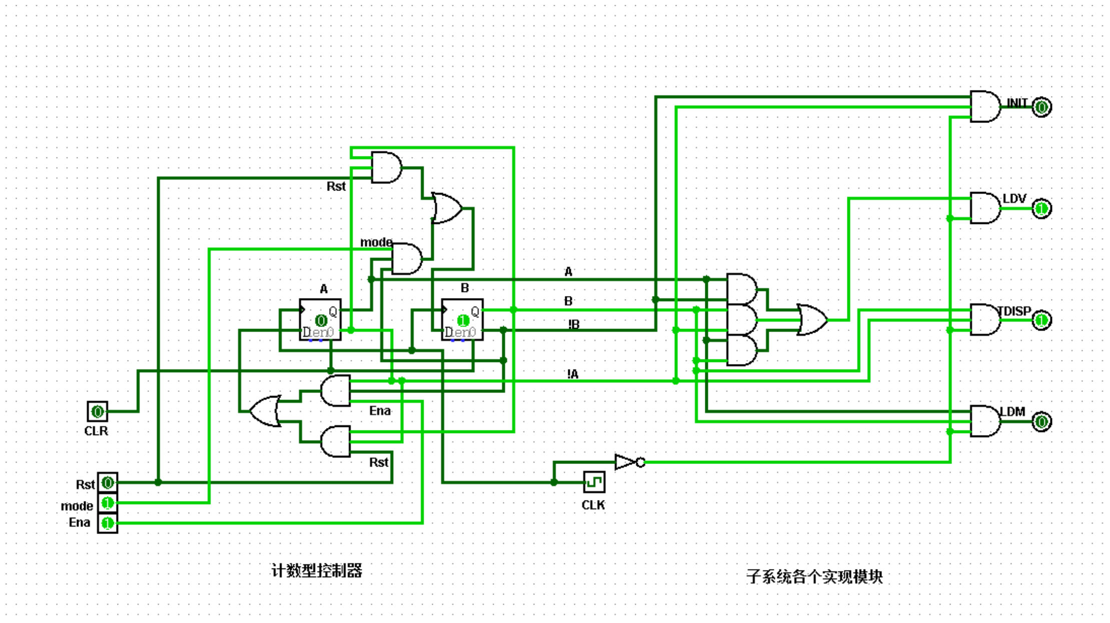
A(D)= $\bar{A}\bar{B}$ ▪Ena+ $\bar{A}B$ $\bullet$ Rst　　　　B(D)= $A\bar{B}\bullet \mathrm{mod}\, e+\bar{A}B\bullet Rst$

控制命令表达式：

INIT= $\bar{A}\bar{B}\bullet T_2$　　　　LDV= $(A\bar{B}+\bar{A}B+AB)\bullet T_2$

TDISP= $B\bar{A}\bullet T_2$　　　　LDM= $AB\bullet T_2$

logisim 模拟



四、子系统模块建模

本系统共划分了六个子系统模块，分别为控制器子系统模块、ADT7420 温度感应子系统模块、数码显示管输出显示子系统模块、VGA 输出显示子系统模块、LED 输出显示子系统模块、蜂鸣器输出音乐模块。具体实现过程又调用了之前已经写过的分频模块和调用 IP 核分频和 ROM 内存。

一、ADT7420 温度传感器模块&&数码管输出显示模块&&LED 输出模块

ADT7420 模块示意图：

FUNCTIONAL BLOCK DIAGRAM

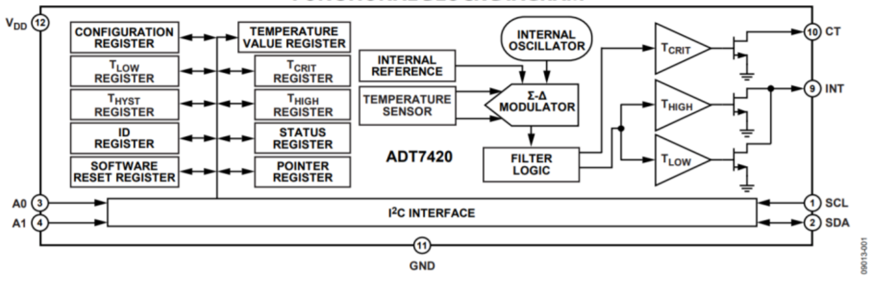
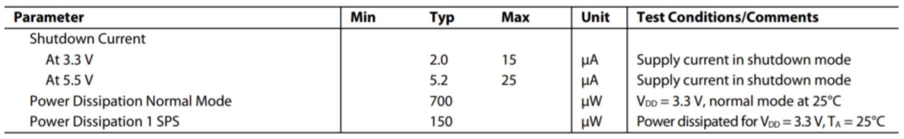


Figure 1.



Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
Shutdown Current					
At 3.3 V		2.0	15	μA	Supply current in shutdown mode
At 5.5 V		5.2	25	μA	Supply current in shutdown mode
Power Dissipation Normal Mode		700		μW	V_{DD} = 3.3 V, normal mode at 25°C
Power Dissipation 1 SPS		150		μW	Power dissipated for V_{DD} = 3.3 V, T_A = 25°C

时序逻辑：

Timing Diagram

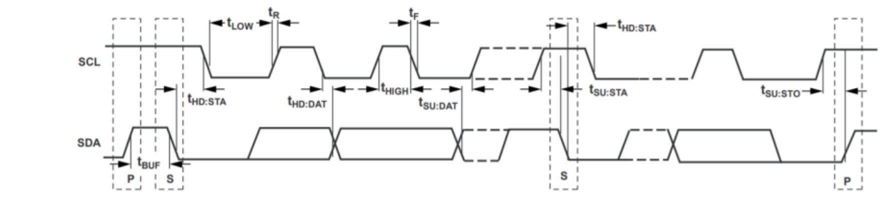


Figure 2. Serial Interface Timing Diagram

接口信号定义：

接口名称	接口属性	接口描述
system_clk	input	系统时钟 100MHZ
ADT7420_SDA	inout	数据传输总线
ADT7420_SCL	output	给 ADT7420 的时钟线
adt_tem	output	采样的 16 位温度
seg	output	数码管的段选信号
an	output	数码管的位选信号
light	output	LED 信号

Verilog 代码:

```
module ADT7420_display(
	input wire system_clk,
	inout wire ADT7420_SDA,
	output wire ADT7420_SCL,
	output wire [15:0] adt_tem,
	output reg [7:0] seg,
	output reg [7:0] an,
	output wire [15:0] light,
	output wire rgbr,
        output wire rgbb
        );
        wire [15:0] temperature;
        assign adt_tem=temperature;
        wire finish_flag;
        reg [5:0] led_display=6'd10;
        ADT7420
right(.system_clk(system_clk),.SCL(ADT7420_SCL),.SDA(ADT7420_SDA),.data(temperature),
.finish_flag(finish_flag),.light(light));
        RGB rgb_led(.system_clk(system_clk),.indata(led_display),.r(rgbr),.b(rgbb));
        wire virtual_clock;
        Divider #(20000)div(.I_CLK(system_clk),.Rst(0),.O_CLK(virtual_clock));
        wire [7:0] segment [0:9];
        assign segment[0]=8'hc0;
        assign segment[1]=8'hf9;
        assign segment[2]=8'ha4;
        assign segment[3]=8'hb0;
        assign segment[4]=8'h99;
        assign segment[5]=8'h92;
        assign segment[6]=8'h82;
        assign segment[7]=8'hf8;
        assign segment[8]=8'h80;
        assign segment[9]=8'h90;
        integer result=0;
        always@(posedge system_clk)begin
                result<=temperature;
                result=(result>>3)*625;
                led_display=result/10000;
                case (an)
                8'b11111110:seg<=8'hc6;
                8'b11111101:seg<=8'h9c;
                8'b11111011:seg<=segment[(result/10)%10];
                8'b11110111:seg<=segment[(result/100)%10];
                8'b11101111:seg<=segment[(result/1000)%10];
                8'b11011111:begin
                        seg[6:0]<=segment[(result/10000)%10][6:0];
                        seg[7]=0;
                end
                8'b10111111:seg<=segment[(result/100000)%10];
                8'b01111111:begin
                        if(result>=0) seg<=segment[(result/1000000)%10];
                        else seg<=8'hbf;
                end
                default:seg<=8'hff;
        endcase
    end
    always@(posedge virtual_clock)begin
        an<=(an<<1)+1;
        if(an==8'hff) an<=8'hfe;
    end
    parameter tem_limit=25;
    reg judge=1;
    always@(*)begin
        if(led_display>=tem_limit) judge<=0;
        else judge<=1;
    end
Endmodule
```

数码管显示模块和 LED 显示模块

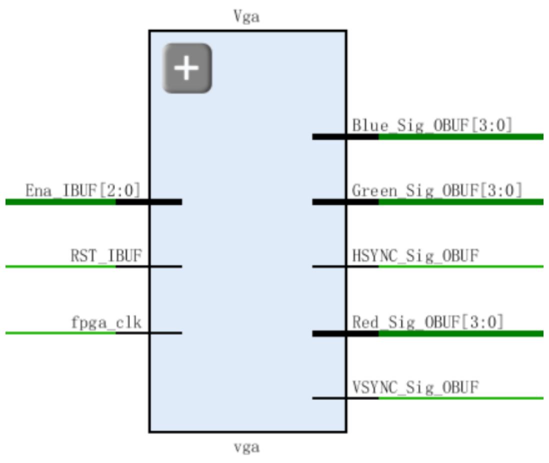
思路：通过数码管的**位选段选**实现输出不同的阿拉伯数字。

verilog 代码：

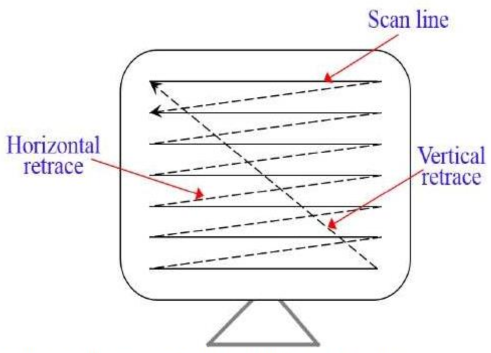
```
module ADT7420_display(
    input wire system_clk,
    inout wire ADT7420_SDA,
    output wire ADT7420_SCL,
    output wire [15:0] adt_tem,
    output reg [7:0] seg,
    output reg [7:0] an,
    output wire [15:0] light,
    output wire rgbr,
    output wire rgbb
    );
    wire [15:0] temperature;
    assign adt_tem=temperature;
    wire finish_flag;
    reg [5:0] led_display=6'd10;
    ADT7420
right(.system_clk(system_clk),.SCL(ADT7420_SCL),.SDA(ADT7420_SDA),.data(temperature),
.finish_flag(finish_flag),.light(light));
    RGB rgb_led(.system_clk(system_clk),.indata(led_display),.r(rgbr),.b(rgbb));
    wire virtual_clock;
    Divider #(20000)div(.I_CLK(system_clk),.Rst(0),.O_CLK(virtual_clock));
    wire [7:0] segment [0:9];
    assign segment[0]=8'hc0;
    assign segment[1]=8'hf9;
    assign segment[2]=8'ha4;
    assign segment[3]=8'hb0;
    assign segment[4]=8'h99;
    assign segment[5]=8'h92;
    assign segment[6]=8'h82;
    assign segment[7]=8'hf8;
    assign segment[8]=8'h80;
    assign segment[9]=8'h90;
    integer result=0;
    always@(posedge system_clk)begin
        result<=temperature;
        result=(result>>3)*625;
        led_display=result/10000;
        case (an)
        8'b11111110:seg<=8'hc6;
        8'b11111101:seg<=8'h9c;
        8'b11111011:seg<=segment[(result/10)%10];
        8'b11110111:seg<=segment[(result/100)%10];
        8'b11101111:seg<=segment[(result/1000)%10];
        8'b11011111:begin
            seg[6:0]<=segment[(result/10000)%10][6:0];
            seg[7]=0;
        end
        8'b10111111:seg<=segment[(result/100000)%10];
        8'b01111111:begin
            if(result>=0) seg<=segment[(result/1000000)%10];
            else seg<=8'hbf;
        end
        default:seg<=8'hff;
        endcase
    end
    always@(posedge virtual_clock)begin
        an<=(an<<1)+1;
        if(an==8'hff) an<=8'hfe;
    end
endmodule
```

二、VGA 显示模块

功能框图：



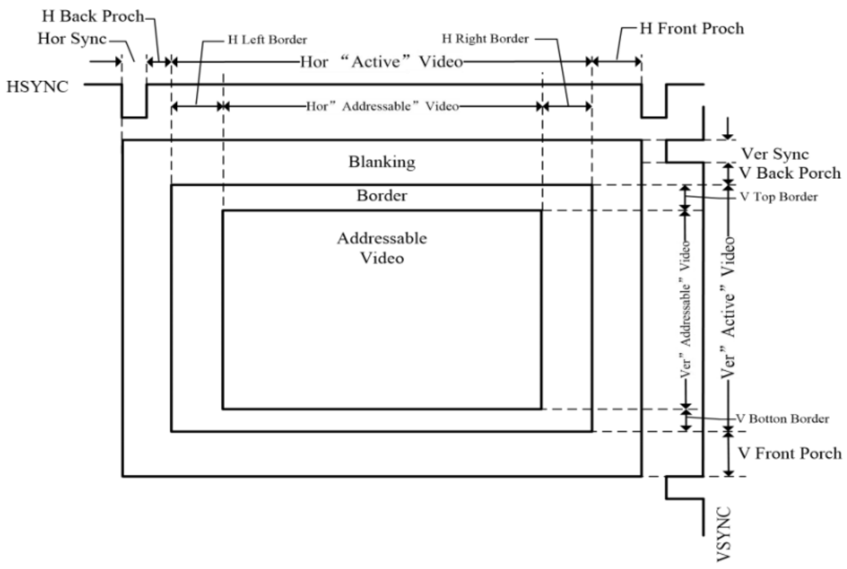
工作原理：



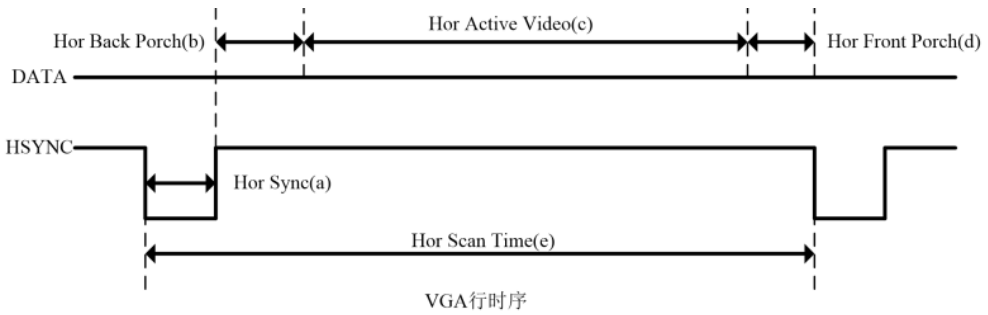
Raster scanning terminology for a single output frame.

VGA 显示器扫描方式从屏幕左上角一点开始，从左向右逐点扫描，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。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，即屏幕的刷新频率。

时序逻辑：

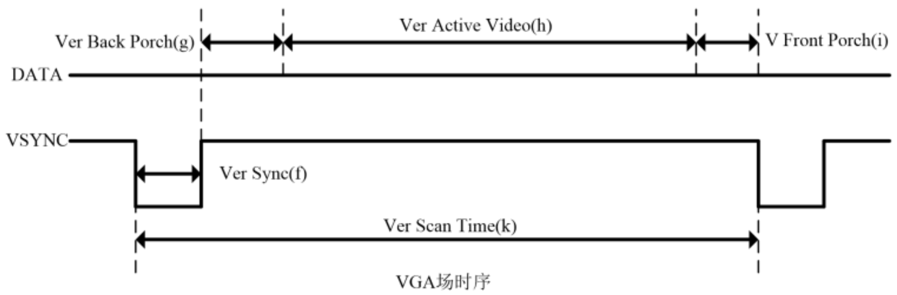


行列扫描



VGA行时序

行扫描



VGA场时序

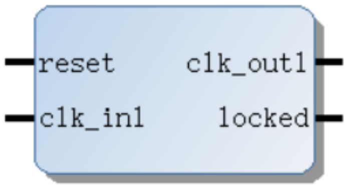
列扫描

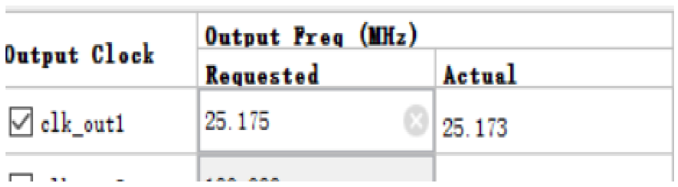
接口信号定义：

接口名称	接口属性	接口描述
system_clk	input	系统时钟 100MHZ
Rst	input	复位信号
ena	input	切换图片的电平信号
VSYNC_Sig	output	列扫描信号
HSYNC_Sig	output	行扫描信号
Red_Sig	output	红色分量信号
Green_Sig	output	绿色分量信号
Blue_Sig	output	蓝色分量信号

此模块调用了 vivado 自带的 IP 核分频和通过 IP 核(block wizard)调用板上的 ROM 内存：

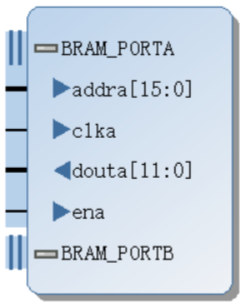
分频：(对于 640*480 的分辨率，对应 25.175MHZ 的频率，实际上 25MHZ 就可以)

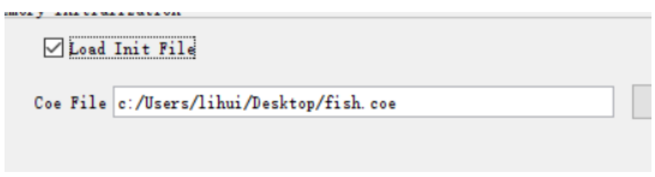




Output Clock	Output Freq (MHz)	
	Requested	Actual
☑ clk_out1	25.175	25.173

Single port ROM：





☑ Load Init File

Coe File c:/Users/lihui/Desktop/fish.coe

verilog 代码：

```
module vga(
        input wire system_clk,
        input wire RST,
        input [2:0] ena,
        output wire VSYNC_Sig,
        output wire HSYNC_Sig,
        output wire [3:0] Red_Sig,
        output wire [3:0] Green_Sig,
        output wire [3:0] Blue_Sig
        );
        vga40HZ fpq(.clk_in1(system_clk),.clk_out1(vga_clk));
        vga_driver driver(.vga_clk(vga_clk),.ena(ena),.rst_n(~RST),.vsync(VSYNC_Sig),
        .hsync(HSYNC_Sig),.red(Red_Sig),.green(Green_Sig),.blue(Blue_Sig));
Endmodule


module vga_driver(//产生行时序，场时序
        input wire vga_clk,
        input wire rst_n,
        input [2:0] ena,//使能信号，切换图片
        output wire vsync,
        output wire hsync,
        output reg [3:0]    red,
        output reg [3:0]    green,
        output reg [3:0]    blue
        );
        parameter image_width =500,
                        image_height=400,
                        image_w=300,
                        image_h=200,
                        image_pix_num=60000;
        reg [15:0] Rom_addr;
        reg [15:0] Rom_addr1;
```

```
	reg [15:0] Rom_addr2;
	wire [11:0] Rom_data;
	wire [11:0] Rom_data1;
	wire [11:0] Rom_data2;
	reg [9:0] Count_H;
	always@(posedge vga_clk or negedge rst_n)begin//行时序
		if(!rst_n)begin
			Count_H<=10'd0;
		end
		else if(Count_H==10'd800-1'b1)begin
			Count_H<=10'd0;
		end
		else begin
			Count_H<=Count_H+1'b1;
		end
	end
	reg [9:0] Count_V;
	always@(posedge vga_clk or negedge rst_n)begin//场时序
		if(!rst_n)begin
			Count_V<=10'd0;
		end
		else if(Count_V==10'd525-1'b1)begin
			Count_V<=10'd0;
		end
		else if(Count_H==10'd800-1'b1)begin
			Count_V<=Count_V+1'b1;
		end
	end
	reg isReady;//产生有效区域标志
	always@(posedge vga_clk or negedge rst_n)begin
		if(!rst_n)begin
			isReady<=1'b0;
		end
		else
if((Count_H>=10'd144&&Count_H<=10'd784)&&(Count_V>=10'd35&&Count_V<=10'd515))b
egin
			isReady<=1'b1;
		end
		else isReady<=1'b0;
	end

	//把 rom 的图片输出
	always@(posedge vga_clk,negedge rst_n)
	begin
```

```
            if(!rst_n)
            begin
               Rom_addr<=16'd0;//!
               Rom_addr1<=16'd0;
            end
            else if(isReady)
                  begin


if((Count_H>=314)&&(Count_H<=314+image_w-1'b1)&&(Count_V>=175)&&(Count_V<=175
+image_h-1'b1))
                       begin
                        if(ena[0]&&~ena[1]&&~ena[2])//1 锦鲤
                                    begin
                                    red<=Rom_data[11:8];
                                    green<=Rom_data[7:4];
                                    blue<=Rom_data[3:0];
                                    if(Rom_addr==image_pix_num-1'b1)
                                           Rom_addr<=16'd0;
                                    else
                                           Rom_addr<=Rom_addr+1'b1;
                                    end
                     else    if(~ena[0]&&ena[1]&&~ena[2])//2 温度计
                       begin
                       red<=Rom_data1[11:8];
                       green<=Rom_data1[7:4];
                       blue<=Rom_data1[3:0];
                       if(Rom_addr1==image_pix_num-1'b1)
                              Rom_addr1<=16'd0;
                       else
                              Rom_addr1<=Rom_addr1+1'b1;
                       end

                       else if(~ena[0]&&~ena[1]&&ena[2])//3 音乐
                                    begin
                                    red<=Rom_data2[11:8];
                                    green<=Rom_data2[7:4];
                                    blue<=Rom_data2[3:0];
                                    if(Rom_addr2==image_pix_num-1'b1)//此处可以改进成
动态效果，下次尝试一下加较小的数字
                                           Rom_addr2<=16'd0;
                                    else
                                           Rom_addr2<=Rom_addr2+1'b1;
                                    end
```

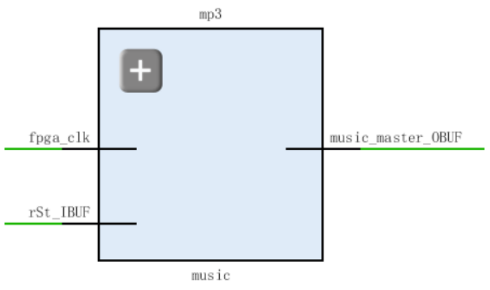
```
                            else
                                                            begin
                                                                red<=4'b1111;
                                                                green<=4'b1111;
                                                                blue<=4'b1111;
                                                                Rom_addr=Rom_addr;
                                                                  Rom_addr1=Rom_addr1;
                                                            end
                        end
                        else
if((Count_H>=144)&&(Count_H<214)&&(Count_V>=35)&&(Count_V<75+image_height-1'b1)
)//左
                        begin
                        red<=4'b1111;
                        green<=4'b0000;
                        blue<=4'b0000;
                        end

                        else
if((Count_H>=214)&&(Count_H<=284+image_width-1'b1)&&(Count_V>=35)&&(Count_V<75
))//上
                        begin
                          red<=4'b0000;
                          green<=4'b1111;
                          blue<=4'b0000;
                        end

                        else
if((Count_H>214+image_width-1'b1)&&(Count_H<=284+image_width-1'b1)&&(Count_V>=75)
&&(Count_V<=115+image_height-1'b1))//右
                        begin
                        red<=4'b0000;
                        green<=4'b0000;
                        blue<=4'b1111;
                        end

                        else
if((Count_H>=144)&&(Count_H<=214+image_width-1'b1)&&(Count_V>=75+image_height-1'b
1)&&(Count_V<=115+image_height-1'b1))//下
                        begin
                        red<=4'b0000;
                        green<=4'b1111;
                        blue<=4'b0000;
                        end
```

```
                    else
                              begin
                                   red<=4'b1111;
                                   green<=4'b1111;
                                   blue<=4'b1111;
                                   Rom_addr=Rom_addr;
                                     Rom_addr1=Rom_addr1;
                              end
          end
          else
               begin
                    red<=4'b1111;
                    green<=4'b1111;
                    blue<=4'b1111;
                    Rom_addr=Rom_addr;
                      Rom_addr1=Rom_addr1;
               end
     end//图片


     assign vsync=(Count_V<=10'd4)?1'b0:1'b1;
     assign hsync=(Count_H<=10'd128)?1'b0:1'b1;
     assign ready=isReady;
     Rom_image rom_image(
     .clka(vga_clk),
     .addra(Rom_addr),
     .ena(ena[0]),
     .douta(Rom_data)
     );

       Rom_image1 rom_image1(
          .clka(vga_clk),
          .addra(Rom_addr1),
          .ena(ena[1]),
          .douta(Rom_data1)
          );
       Rom_image2 rom_image2(
                 .clka(vga_clk),
                 .addra(Rom_addr2),
                 .ena(ena[2]),
                 .douta(Rom_data2)
                 );
Endmodule
```

三、蜂鸣器输出模块

音乐有 8 个音符，每个音符对应一个频率，不同的频率在蜂鸣器上对应不同的时延，从而使蜂鸣器能够发出悦耳的铃声音乐，我找了一首简短的乐谱将其制作成类似于前面图片文件的 coe 文件，同样调用 IP 核 ROM 内存读取音乐文件，要注意不同长度的音符对应不同的宽度，在调用 IP 核时要注意。

功能框图



接口信号定义：

接口名称	接口属性	接口描述
clk_100	input	系统时钟 100MHZ
Rst	input	复位信号
wave	output	蜂鸣器的频率
d12	output	拉高电平信号

verilog 代码：

```
module music(//循环播放音乐
input clk_100,//100mhz
input Rst,
output wave,//蜂鸣器的频率
output d12//拉高
);
wire [2:0] data_note;
wire clk_single;//读取音符的时钟
reg [7:0] number_now;//当前播放的音符序号
reg [31:0] n;//分频系数,生成
parameter num=8'd184;//音符的个数

parameter n_single=32'd25000000;//0;//单个音符持续播放的时间的时钟分频参数
div_ d_signle(clk_100,n_single,Rst,clk_single);//单个音符持续的时间
christmas christmas_(.clka(clk_100),.addra(number_now),.douta(data_note));//获取音符
wire mid_wave;
reg rst;
div_ d_note(clk_100,n,Rst,mid_wave);
```

```
initial
begin
        rst<=1'b0;
        number_now<=8'b0;
end

always @(negedge clk_100)
if(rst==1)
        rst<=0;

assign wave=(n!=32'd0)?mid_wave:16'b0;

always@(posedge clk_single)//读取新音符
begin
        if(number_now==num-1)
                number_now<=8'b0;
        else
                number_now<=number_now+1;
end

//下面是分频常数
always@(data_note)
begin
        case(data_note)
                3'd0:n<=32'd0;
                3'd1:n<=32'd191095;
                3'd2:n<=32'd170270;
                3'd3:n<=32'd151676;
                3'd4:n<=32'd143163;
                3'd5:n<=32'd127551;
                3'd6:n<=32'd113636;
                3'd7:n<=32'd101235;
        endcase
        rst<=1;
end
assign d12=1'b1;
Endmodule
```

四、控制器子系统模块（顶层控制）

接口信号定义：

接口名称	接口属性	接口描述
fpga_clk	input	系统时钟 100MHZ
mode	input	模式切换

Rst	input	系统复位信号
rSt	input	音乐启动信号
Ena	input	图片切换信号
light	output	LED 信号
seg	output	数码管的段选信号
an	output	数码管的位选信号
music_master	output	蜂鸣器频率信号
d12	output	电平拉高信号
ADT7420_SCL	output	给 ADT7420 的时钟线
ADT7420_SDA	output	数据传输总线
Red_Sig	output	红色分量信号
Green_Sig	output	绿色分量信号
Blue_Sig	output	蓝色分量信号

verilog 代码：

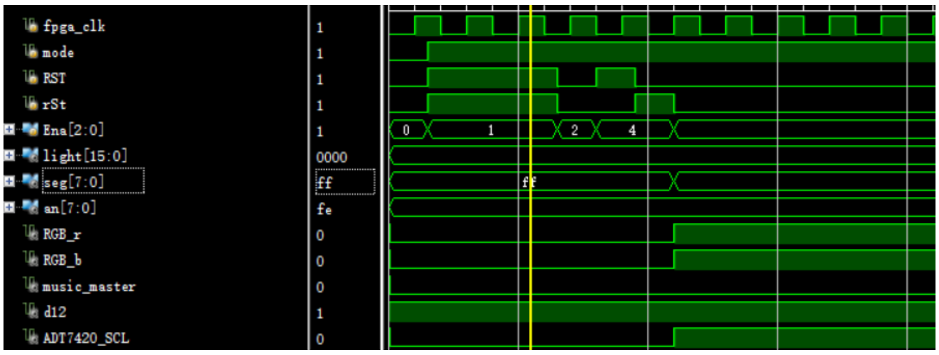
```
module master(
        input wire fpga_clk,//系统时钟
        input wire mode,
        input wire RST,//复位信号
        input rSt,
        input [2:0] Ena,
        output wire [15:0] light,
        output wire [7:0] seg,
        output wire [7:0] an,
        output wire RGB_r,
        output wire RGB_b,
        output music_master,
        output d12,
        inout wire ADT7420_SDA,
        output wire ADT7420_SCL,
        output wire VSYNC_Sig,
        output wire HSYNC_Sig,
        output wire [3:0] Red_Sig,
        output wire [3:0] Green_Sig,
        output wire [3:0] Blue_Sig
        );
        wire [7:0] Aseg;
        wire [7:0] Aan;
        wire [15:0] Alight;
        wire Argbr;
        wire Argbb;
        wire [7:0] Dseg;
```

```
	wire [7:0] Dan;
	wire [15:0] Dlight;
	wire [15:0] adt_tem;//温度
	wire MODE;
	Anti_shake mode_change(.clk_in(fpga_clk),.key_in(mode),.key_out(MODE));
	ADT7420_display adt(.system_clk(fpga_clk),.ADT7420_SDA(ADT7420_SDA),
	.ADT7420_SCL(ADT7420_SCL),.seg(Aseg),.an(Aan),.light(Alight),
	.rgbr(Argbr),.rgbb(Argbb),.adt_tem(adt_tem));

	music mp3(.clk_100(fpga_clk),.Rst(rSt),.wave(music_master),.d12(d12));

	vga Vga(.system_clk(fpga_clk),.ena(Ena),.RST(RST),.VSYNC_Sig(VSYNC_Sig),
	.HSYNC_Sig(HSYNC_Sig),.Red_Sig(Red_Sig),.Green_Sig(Green_Sig),
	.Blue_Sig(Blue_Sig));
	assign light=MODE?Dlight:Alight;
	assign seg=MODE?Dseg:Aseg;
	assign an=MODE?Dan:Aan;
Endmodule
```

五、测试模块建模

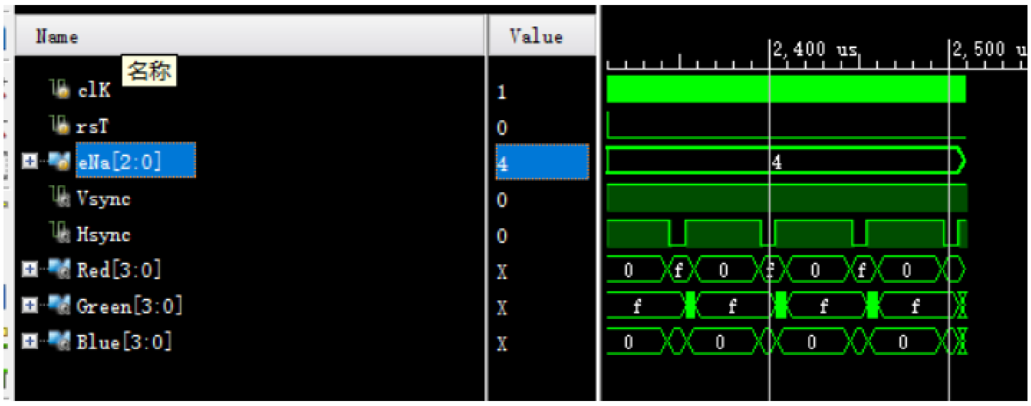
一、控制器子系统模块测试

```
module master_tb();
reg fpga_clk;
reg mode;
reg RST;
reg rSt;
reg [2:0]Ena;
wire [15:0]light;
wire [7:0]seg;
wire [7:0]an;
wire RGB_r;
wire RGB_b;
wire music_master;
wire d12;
wire ADT7420_SCL;
wire ADT7420_SDA;
wire Red_sig;
wire Blue_sig;
wire Green_sig;
wire HSYNC_sig;
wire VSYNC_sig;

Final_Design master_tb(.fpga_clk(fpga_clk),.mode(mode),.RST(RST),.rSt(rSt),.Ena(Ena),
.light(light),.seg(seg),.an(an),.RGB_r(RGB_r),.RGB_b(RGB_b),.music_master(music_master),
.d12(d12),.ADT7420_SCL(ADT7420_SCL),.ADT7420_SDA(ADT7420_SDA),.Red_Sig(Red_si
g),
.Blue_Sig(Blue_sig),.Green_Sig(Green_sig),.HSYNC_Sig(HSYNC_sig),.VSYNC_Sig(VSYNC_s
ig)
);

initial
begin
fpga_clk=0;
mode=0;
RST=0;
rSt=0;
Ena=3'b000;
#15
mode=1;
RST=1;
rSt=1;
Ena=3'b001;
#50
RST=0;
rSt=0;
Ena=3'b010;
#15
RST=1;
Ena=3'b100;
#15
RST=0;
rSt=1;
#15
RST=0;
rSt=0;
Ena=3'b000;
end
always
#10 fpga_clk=~fpga_clk;
Endmodule
```



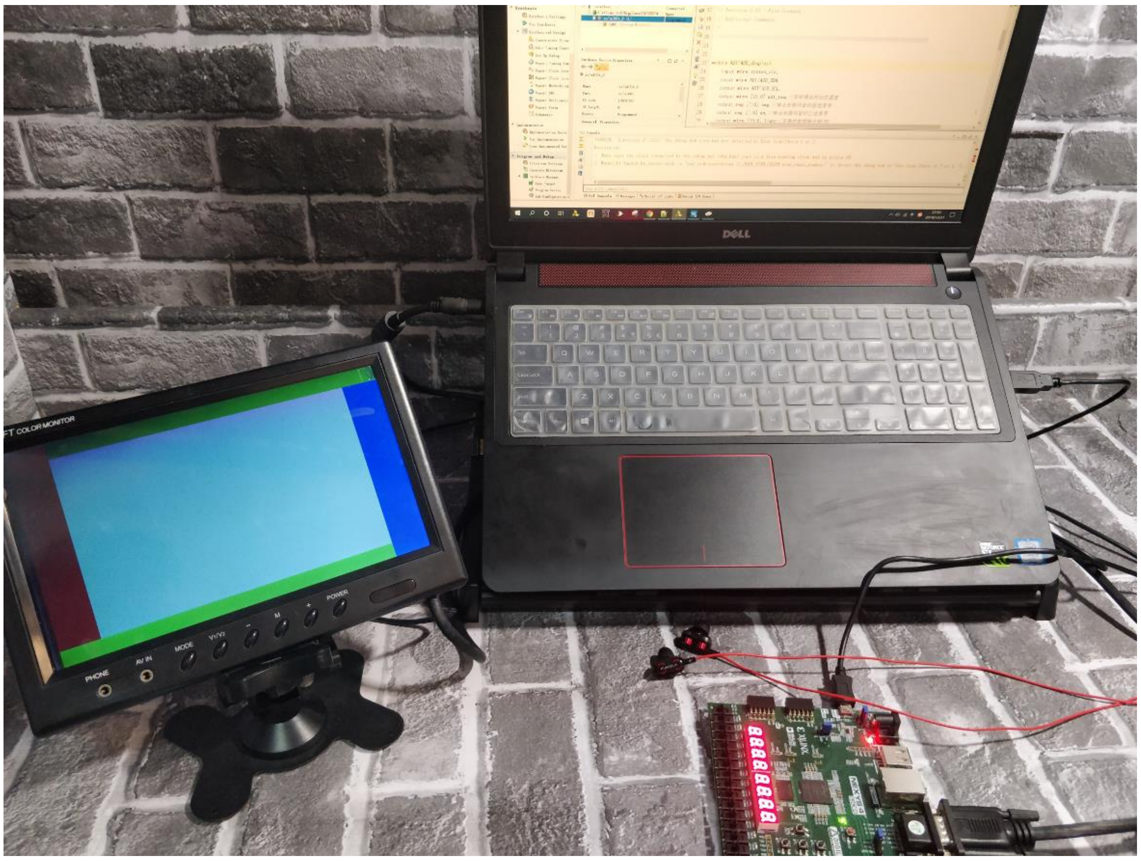
二、vga 显示模块测试

```
module vga_tb();
  reg system_clk;//时钟信号，上升沿有效
  reg RST_n;//复位信号，低电平有效
reg [2:0] ena;
  wire Vsync_s;//行同步信号
  wire Hsync_s;//列同步信号
  wire [3:0]R_s;//红颜色信号
  wire [3:0]G_s;//绿颜色信号
  wire [3:0]B_s;//蓝颜色信号

  initial//时钟
  CLK = 0;
  always
  #1 CLK = ~CLK;

  initial//复位信号
  begin
  RST_n = 0;
  #1
  RST_n = 1;
  end
  //实例化测试模块
  VGA
uut(.system_clk(system_clk), .RST(RST_n), .ena(ena),.VSYNC_Sig(Vsync_s), .HSYNC_Sig(Hs
ync_s), .Red_Sig(), .Green_Sig(G_s), .Blue_Sig(B_s));
Endmodule
```


Name
Value
名称
clK
rsT
eNa[2:0]
Vsync
Hsync
Red[3:0]
Green[3:0]
Blue[3:0]
2,400 us
2,500 u

六、下板测试

1、复位



2、显示图片

3、测量温度（室温）

4、音乐播放