八、4.3 寸 RGB 显示屏控制器设计

各种常见显示屏极其接口介绍

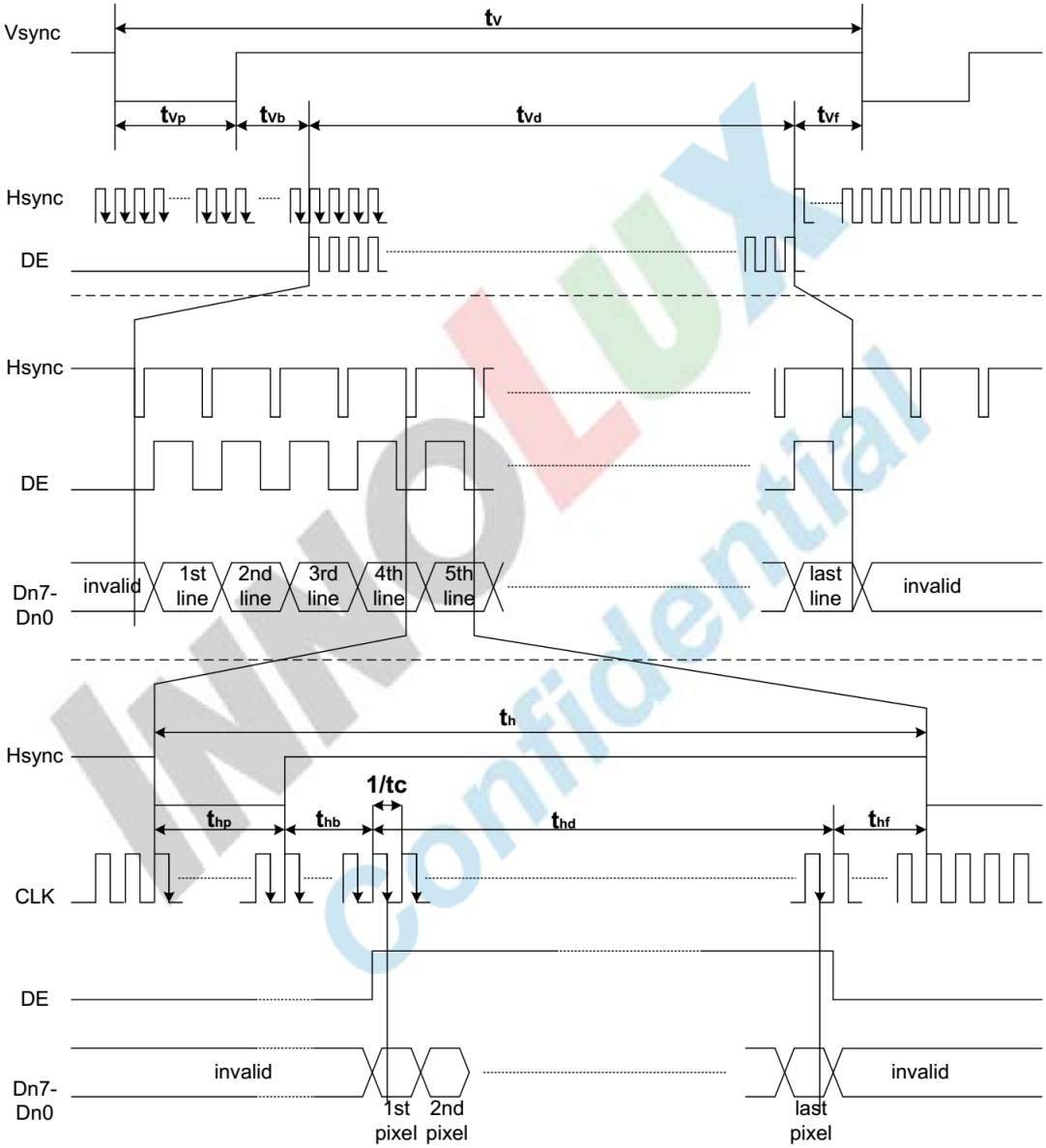
在嵌入式系统中，经常会使用到液晶显示屏来显示相关信息，从早期的 1602 字符点阵到如今的 4K LED 显示屏，显示技术经历了巨大的变革。根据应用领域不同，也有很多不同的屏幕接口，如 8080 接口、RGB 接口、MIPI 接口等，每种接口都有各自的特性，应用于不同的场合，交叉较少。我们在使用时，也是需要根据实际需求合理选择合适接口的显示屏。

相信很多电子生刚开始学习单片机时，接触到的就是 51 单片机。在 51 单片机上，使用最广泛的显示设备就是 1602、12864 字符点阵。这两种液晶显示屏使用的是 Intel 8080 的总线， 51 单片机使用外扩 8080 总线或者 IO 模拟该总线对液晶屏进行读写，以实现数据显示。

后来，当我们学习了较为高级的单片机，如 MSP430、STM32 后，我们接触到了彩色液晶显示屏。单片机经常使用的彩色液晶显示屏通常都是内部设计了有显示缓存存储器的，只要把要显示的内容写入到显存中，显示屏就会自动的依次读取每个像素点对应的显存内容并驱动液晶像素点显示对应颜色。彩色液晶显示屏模型和 12864 液晶屏较为类似，只不过能够显示彩色图像，而 12864 只能显示黑白内容。这类彩色液晶显示屏常使用 16 位的总线接口，该接口也与 Intel 8080 总线接口兼容，我们在使用的时候，只需要将彩色液晶显示屏当做一个 RAM 进行读写即可将需要显示的内容显示在屏幕对应的位置。比较典型的一种使用方法，是使用 STM32 的 FSMC 总线，将这样的液晶显示屏映射为一块存储区，直接使用 FSMC 总线对该存储区进行读写即可。

随着学习的进一步深入，部分同学接触到了嵌入式 ARM 处理器，如 ARM9、ARM11、Cortex-A8 等等，这些处理器往往运行成熟的操作系统，如 WindowsCE、Linux 等等。而且它们都能驱动很多的大屏幕，如市面上最早流行的 2440 的开发板，基本标配都有一块 4.3 寸 480*272 分辨率的 RGB 显示屏，而有些甚至支持到了 7 寸 800*480 分辨率。这些屏幕，使用普通的 MCU 是无法驱动的，因为这些屏幕本身不含显存，需要驱动器能够带有显存，并按照 RGB 时序准确的将显存中的数据送到屏幕上显示，而一般的单片机由于工作速度、总线带宽、存储容量有限，很难支持如此高的数据刷新速率。而 ARM 处理器由于性能较高，而且使用外部大容量 DDR 存储器作为运行内存，因此能够支持这样一类的屏幕。

以下为一款常用的 4.3 寸 TFT 显示屏（AT043TN24 V.1）的接口时序图：



相信之前学习过 VGA 时序的朋友一眼就能看出来，是的，这款屏幕的接口和 VGA 接口时序非常的相似，甚至说是一样的。区别只是在于时间参数有所不同。看到这里，相信大家心里瞬间就有了信心，VGA 都能够理解，这样一个和 VGA 接口极度相似的接口，又有什么搞不定的呢？而事实就是，我们后面设计该显示屏的驱动的时候，就只是在之前 VGA 控制器的基础上进行了简单的一些时序参数的修改，就能够直接用来驱动这样一款 4.3 寸 TFT 液晶显示屏了。

4.3 寸 RGB 接口 TFT 显示模组介绍：

市面上我们最常买到的 4.3 寸屏都是以模组的形式提供的，各大厂家生产的模组，其接口引脚顺序都是兼容的。而且接口时序也都是标准的。以下为 AT043TN24 V.1 这个模组的接口顺序：

引脚编号	引脚名	引脚功能说明
1	V_{LED-}	LED 背光灯供电阴极
2	V_{LED+}	LED 背光灯供电阳极
3	GND	TFT 工作电源地
4	VDD	TFT 工作电源 VCC
5~12	R0~R7	红色分量数据 0~7 位
13~20	G0~G7	绿色分量数据 0~7 位
21~28	B0~B7	蓝色分量数据 0~7 位
29	GND	TFT 工作电源地
30	PCLK	TFT 像素时钟
31	DISP	显示开/关
32	HSYNC	行同步信号
33	VSYNC	场同步信号
34	DE	数据使能信号
35	NC	悬空
36	GND	TFT 工作电源地
37	X1	差分模拟触摸接口右侧电极
38	Y1	差分模拟触摸接口底侧电极
39	X2	差分模拟触摸接口左侧电极
40	Y2	差分模拟触摸接口上方电极

其中，1、2 号脚为背光灯的供电脚，TFT 屏背光使用串联 LED 供电，要求供电电压为 18.6V 到 21V，恒流，电流范围为 36mA 到 44mA 之间。

3、4、29、36 号脚为 TFT 的供电电源或 GND 脚，用于给 TFT 提供必要的工作电压。

5~12、13~20、21~28 分别为像素颜色数据的红绿蓝分量输入脚，每个颜色分量共 8 位，即该液晶显示屏最高可现实 2^24 中颜色。

30 脚为像素时钟脚，驱动器与 TFT 屏的数据和控制信号全部需要同步于该像素时钟信号。

31 脚为现实开关脚，使用该脚可直接控制 TFT 屏的显示开关。

32、33 为行、场同步信号，实现对图像数据的行、列同步。用来控制 TFT 屏的刷新。这些信号与 VGA 时序含义完全一致，因此只要理解了 VGA 接口原理。

34 脚为数据使能脚，即在显示有效区域，打开该信号以使能信号输入，在非有效区域，关闭该信号以禁止像素数据输入，以免影响到消隐。

37~40 脚为 4 线制电阻触摸接口，在设计含有触摸功能的显示组件时，常把触摸屏的 4 根信号线先焊接到 TFT 屏的 FPC 对应的焊盘上，然后通过 TFT 屏的这 4 个脚送往触摸控制器（如 XPT2046）。当然，也可以不使用这 4 个脚，而是直接将触摸屏的 4 根连线直接连接到触摸控制器上，这取决于应用需求。

RGB 接口 TFT 屏扫描方式

RGB 接口的 TFT 屏扫描方式和 VGA 类似，也是使用行列扫描的方式。在介绍 TFT 屏扫描原理之前，我们还是再来回顾下 VGA 显示器的扫描原理。

在 VGA 标准兴起的时候。常见的彩色显示器一般由 CRT（阴极射线管）构成，色彩是

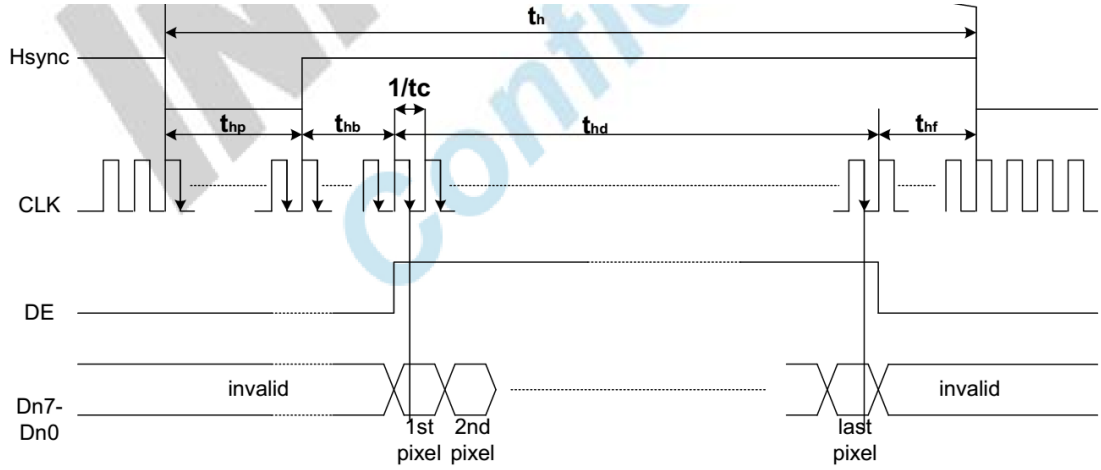
由 RGB（红、绿、蓝）三基色组成。显示是用逐行扫描的方式解决。阴极射线枪发出电子束打在涂有荧光粉的荧光屏上，产生 RGB 三基色，合成一个彩色像素，扫描从屏幕的左上方开始，从左到右，从上到下进行扫描，每扫完一行，电子束都回到屏幕的左边下一行的起始位置。在这期间，CRT 对电子束进行消隐。每行结束时，用行同步信号进行行同步；扫描完所有行，用场同步信号进行场同步，并使扫描回到屏幕的左上方。同时进行场消隐，预备下一场的扫描。

随着显示技术的发展，出现了液晶显示器，液晶显示器的成像原理与 CRT 不同，液晶显示器是通过对液晶像素点单元施加电压与否，来实现液晶单元的透明程度，并添加三色滤光片、分别使 R、G、B 这 3 中光线透过滤光片，最后通过 3 个像素点合成一个彩色像素点，从而实现彩色显示。但是由于液晶显示技术后于 CRT 显示技术诞生，因此在液晶显示器诞生的时候，为了能够兼容传统的显示接口，因此液晶显示器通过内部电路实现了对 VGA 接口的完全兼容。因此，我们在使用显示器时，只要该显示器带有标准的 VGA 接口，我们就不用去关系其成像原理，直接使用标准的 VGA 时序即可驱动。

而 RGB 接口的 TFT 屏，扫描方式与 VGA 完全一致，不同之处只是在于，VGA 显示器是接收模拟信号，而 TFT 屏则省略了这一过程，直接接受数字信号。例如，驱动 VGA 时，我们首先产生对应像素的颜色数字信号编码，然后使用数模转换电路将数字转换为模拟信号，然后通过 VGA 线缆将模拟信号传输到 VGA 显示器上进行显示。而 TFT 屏则直接省略了数模转换这一过程，直接接受传输线缆传输过来的数字信号，并进行显示。因此在控制器设计端，并没有任何区别。

4.3 寸 RGB 显示屏时序分析

上一节我们讲了，RGB 接口的 TFT 屏其驱动时序与 VGA 非常类似，而区别仅仅在于时序参数不同，接下来我们就来看下 TFT 屏的时序参数。下图是该屏接口的行扫描、场扫描的时序图。



行扫描时序要求（单位：输出一个像素的时间间隔，即像素时钟）：

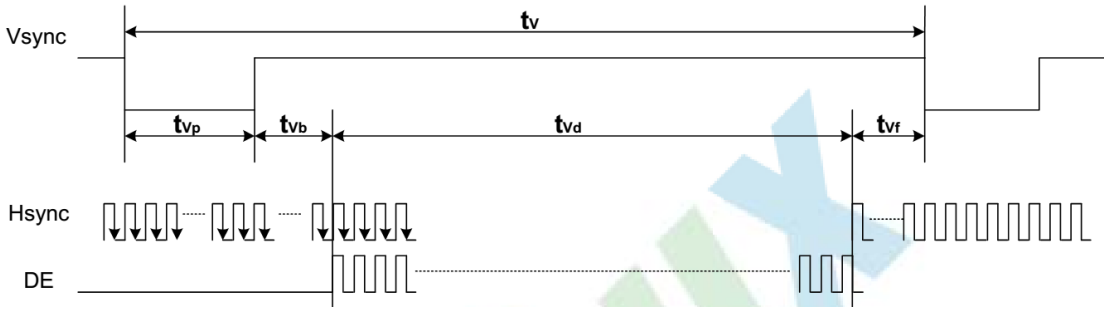
T_{hp}（行同步头）：41

T_{hb}：2

T_{hd}（行图像）：480

T_{hf}：2

T_h：525



场扫描时序要求（单位：输出一行 Line 的时间间隔）：

T_{vp}（场同步头）：10

T_{vb}：2

T_{vd}（场图像）：272

T_{vf}：2

T_v：286

类比 VGA 工业标准，480*272 分辨率 RGB 屏标准所要求的频率如下：

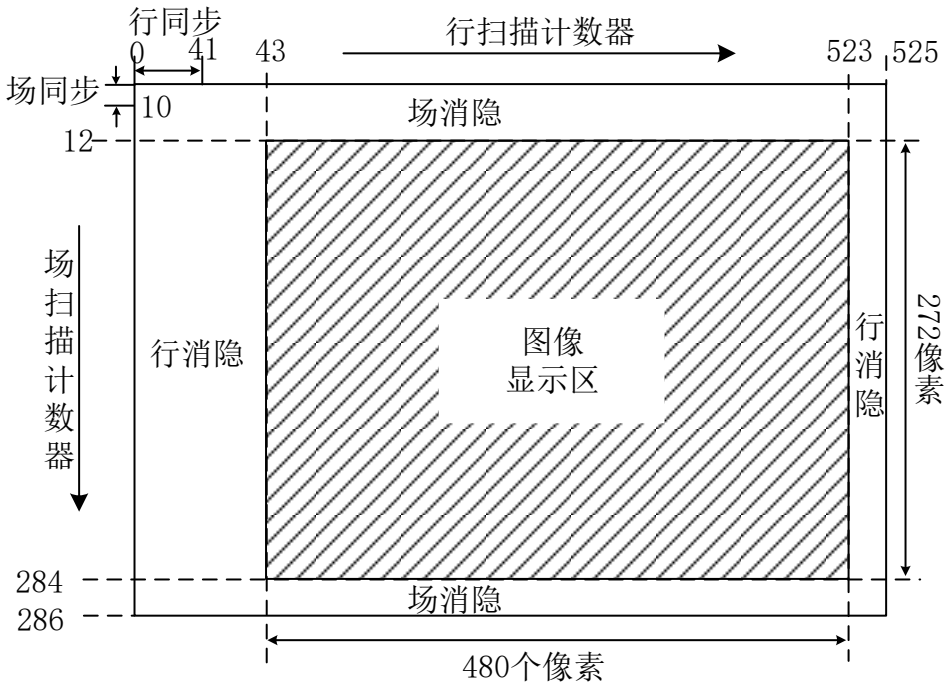
时钟频率 9MHz（像素输出的频率）

行频率 31469 Hz

场频率 59.94Hz（每秒图像刷频率）

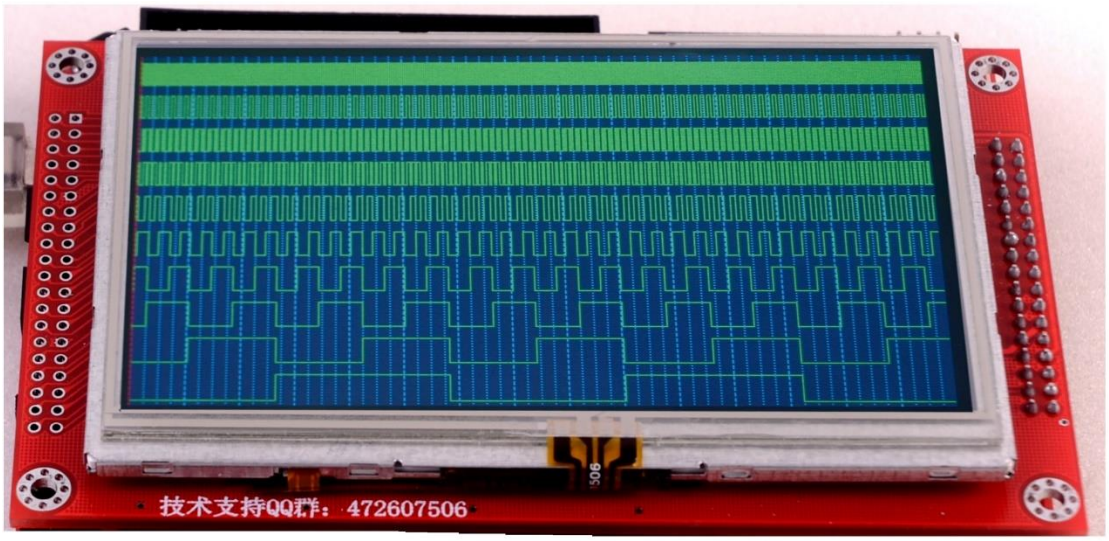
同 VGA 工业标准显示模式类似，RGB 接口的行同步、列同步也都为负极性，即同步脉冲要求是负脉冲。

下图为 480*272 分辨率 RGB 屏的图像显示扫描示意图，在设计时，可用两个计数器进行计数（行、场扫描计数器），行计数器的驱动时钟为 9MHz，场计数器的驱动时钟为行计数器的溢出信号。计数的同时控制行、场同步信号输出。并在适当的时候送出数据，就能显示相应的图像。注意消隐器件送出的数据应该为 0x0000。显示器的刷新频率为 9MHz/525/286 = 59.94Hz，等于 VGA 工业标准场帧频 59.94Hz



芯航线 FPGA 学习套件 VGA 电路介绍

芯航线 FPGA 学习套件提供了一款 4.3 寸 TFT 触摸显示组件，该组件使用了一片 AT043TN24 模组，并安装了 4 线制触摸屏。下图为芯航线 4.3 寸 TFT 触摸显示组件示意图：



本模组实现了对电阻触摸屏的模拟信号采样电路。并将 TFT 屏的 24 位颜色数据转化为 16 位数据，然后通过 34 针排针接口，最后与芯航线 FPGA 主板相连。

在前面介绍中提到，该屏幕的颜色数据支持 24 位输入，即每种颜色有 8 位表示。而我们常见的数字系统，常使用 16 位色进行显示图像，因此，这里为了节约 FPGA 引脚，只选取了 24 位颜色数据中的 16 根数据线。为了保证精简数据线后，图像颜色能够不失真，因此分别选取 R、G、B 数据线的高 5、6、5 位，组成新的 16 位数据线，屏 24 位数据线和芯航线 TFT 触摸显示屏组件 16 位数据线对应关系如下所示：

TFT24 位数据线	R[7:3]	R[2:0]	G[7:2]	G[1:0]	B[7:3]	B[2:0]
模组 16 位数据线	TFT_RGB [15:11]	NC	TFT_RGB [10:5]	NC	TFT_RGB [4:0]	NC

芯航线 4.3 寸触摸显示组件 RGB 接口三基色信号 R、G、B 共用 16 位（分别是 R 为 5 位、G 为 6 位、B 为 5 位），因此可以显示 65536 种颜色。RGB 数据的格式如下表所示：

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D1
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

以下为常见的几种颜色对应的数据编码：

颜色	黑	蓝	红	紫	绿	青	黄	白
R	0	0	1	1	0	0	1	1
G	0	0	0	0	1	1	1	1
B	0	1	0	1	0	1	0	1
数据编码	0x0000	0x001F	0XF800	0XF81F	0x07E0	0x07FF	0XFFE0	0xFFFF

小结

通过以上介绍，我们了解了实现 RGB 接口屏驱动的行列扫描方法，即使用两个计数器分别进行行、场计数，根据计数值确定像素数据内容和行、场同步信号的电平状态。同时，也知道了要显示不同的颜色，只需要给 D0~D15 不同的数据，即可显示不同的颜色。

RGB TFT 控制器设计

第一步，设计行扫描计数器

行扫描计数器即每个像素时钟自加 1，一旦加满到 524（刚好 525 个时钟周期），计数器清零并重新技术，该部分代码可如下设计：

```
reg [9:0] hcount_r;     //TFT 行扫描计数器

//**********************TFT 驱动部分**********************
//行扫描计数器
always@(posedge Clk9M or negedge Rst_n)
if(!Rst_n)  //复位时，让行扫描计数器清零
    hcount_r<=10'd0;
else if(hcount_r==10'd524)  //当一行数据扫完后，再次清零行扫描计数器
    hcount_r<=10'd0;
else    //0~524 之间，每个像素时钟时行扫描计数器自加 1
    hcount_r<=hcount_r+10'd1;
```

第二步，设计场扫描计数器

由于场扫描计数器是在每次一行扫描完成后加 1 的，即场扫描计数器的自加条件是行扫描计数器溢出。所以，场扫描计数器的自加条件为行扫描完成，即“hcount_r==10'd524”，场扫描计数器代码如下所示：

```
reg [9:0] vcount_r;     //TFT 场扫描计数器

//场扫描
always@(posedge Clk9M or negedge Rst_n)
if(!Rst_n)  //复位时让场计数器清零
    vcount_r<=10'd0;
else if(hcount_r==10'd524) begin    //每次一行扫描完成
    if(vcount_r==10'd285)   //每次一场扫描结束，清零计数器
        vcount_r<=10'd0;
    else
        vcount_r<=vcount_r+10'd1;//场计数器在 0~285，满足条件，自加 1
end
else //不满足行扫描结束条件器件，让场扫描计数器保持不变
    vcount_r<=vcount_r;
```

第三步，产生行同步信号和场同步信号

根据 480*272 分辨率 RGB 屏接口标准时序，我们知道每一个完整的帧都包含了数据段和消隐段，在消隐段期间，行同步信号和列同步信号有一段行同步头和场同步头，在同步期间，对应行同步信号或者场同步信号为低电平，因此我们可以根据行、场计数器的值来确定行、场同步信号的电平状态。对于行同步信号，其行同步头为一行扫描的前 41 个像素时钟周期，因此行同步信号可用如下的简单方式控制：

```
assign TFT_HS=(hcount_r>10'd40);
```

对于场同步信号，其场同步头为一行扫描的前 10 个像素时钟周期，因此行同步信号可用如下的简单方式控制：

```
assign TFT_VS=(vcount_r>10'd9);
```

第四步，输出数据

TFT 控制器的设计目的是为了驱动 TFT 显示屏显示需求的图像内容，因此需要设计数据输出部分，这里，数据来源可以为其它部分产生的图像信号，如摄像头数据、BMP 图片数据。我们在驱动 TFT 时，只需要保证在扫描正确的像素点时，其它部分产生的图像信号能够与该像素点位置对应上，则不需要对图像数据再进行二次处理，但是，在行、场消隐期间，需要保证输出到 TFT 屏的 RGB 数据线上的数据全部为 0，因此可以设置一个二选一多路器，只有在非消隐期间，TFT 控制器才直接输出其他部分输入的图像数据，而消隐器件则强制输出全 0。

我们可以首先产生一个图像数据有效标志信号，然后使用该标志信号控制 TFT 输出数据的内容，即切换二选一多路器的通道，从而实现消隐器件数据全 0 的功能。

图像数据有效标志信号产生代码如下所示：

```
    //数据、同步信号输出
    assign dat_act=((hcount_r>=10'd42)&&(hcount_r<10'd522))
                  &&((vcount_r>=10'd11)&&(vcount_r<10'd283));
```

dat_act 即为图像数据有效标志信号。

消隐强制输出 0 二选一多路器代码如下所示：

```
    assign TFT_RGB=(dat_act)?data_in:16'h0000;
```

其中，TFT_RGB 是输出到 VGA 接口上的数据，而 data_in 则是其他模块传递过来的正确的图像数据。

第五步，输出正确的行列扫描位置

为了使其他模块能够根据当前扫描位置正确的输出图像数据，因此需要将 TFT 控制器的实时扫描位置输出，以供其他模块使用。

```
    assign hcount=hcount_r-10'd42;
    assign vcount=vcount_r-10'd11;
```

完整 TFT 控制器设计

以上为我们根据直观思维设计的驱动电路，在代码中，直接使用了数字作为运算和比较的内容，这样不利于修改。因此，为了实现易于修改的控制器设计，方便后期简单修改后兼容其他分辨率，对代码进行优化，使用参数化设计。将代码中使用到的一些与时序相关的数字直接使用 parameter 这样的参数进行定义，这样在以后需要修改时间参数时，只需要修改 parameter 定义的内容即可，不需要再深入到代码中一个一个修改。这里不再一一介绍如何修改，只贴出最终设计修改完成的代码，请用户自行比对领悟。

```
module TFT_CTRL(
    Clk9M,  //系统输入时钟 9MHZ
    Rst_n,  //复位输入，低电平复位
    data_in,    //待显示数据
    hcount,     //TFT 行扫描计数器
    vcount,     //TFT 场扫描计数器
    TFT_RGB,    //TFT 数据输出
    TFT_HS,     //TFT 行同步信号
    TFT_VS,     //TFT 场同步信号
    TFT_CLK,        //TFT 像素时钟
    TFT_DE,     //TFT 数据使能
    TFT_PWM     //TFT 背光控制
);

    //-----------------模块输入端口-----------------
```

```
    input  Clk9M;          //系统输入时钟 9MHZ
    input  Rst_n;
    input  [15:0]data_in;     //待显示数据

    //-----------------模块输出端口----------------
    output [9:0]hcount;
    output [9:0]vcount;
    output [15:0]TFT_RGB;  //TFT 数据输出
    output TFT_HS;          //TFT 行同步信号
    output TFT_VS;          //TFT 场同步信号
    output TFT_CLK;
    output TFT_DE;
    output TFT_PWM;

    //-----------------内部寄存器定义----------------
    reg [9:0] hcount_r;     //TFT 行扫描计数器
    reg [9:0] vcount_r;     //TFT 场扫描计数器
    //-----------------内部连线定义----------------
    wire hcount_ov;
    wire vcount_ov;
    wire dat_act;//有效显示区标定

    //TFT 行、场扫描时序参数表
    parameter TFT_HS_end=10'd40,
              hdat_begin=10'd42,
              hdat_end=10'd522,
              hpixel_end=10'd524,
              TFT_VS_end=10'd9,
              vdat_begin=10'd11,
              vdat_end=10'd283,
              vline_end=10'd285;

    assign hcount=hcount_r-hdat_begin;
    assign vcount=vcount_r-vdat_begin;

    assign TFT_CLK = Clk9M;
    assign TFT_DE = dat_act;
    assign TFT_PWM = Rst_n;

    //**********************TFT 驱动部分**********************
    //行扫描
    always@(posedge Clk9M or negedge Rst_n)
    if(!Rst_n)
        hcount_r<=10'd0;
```

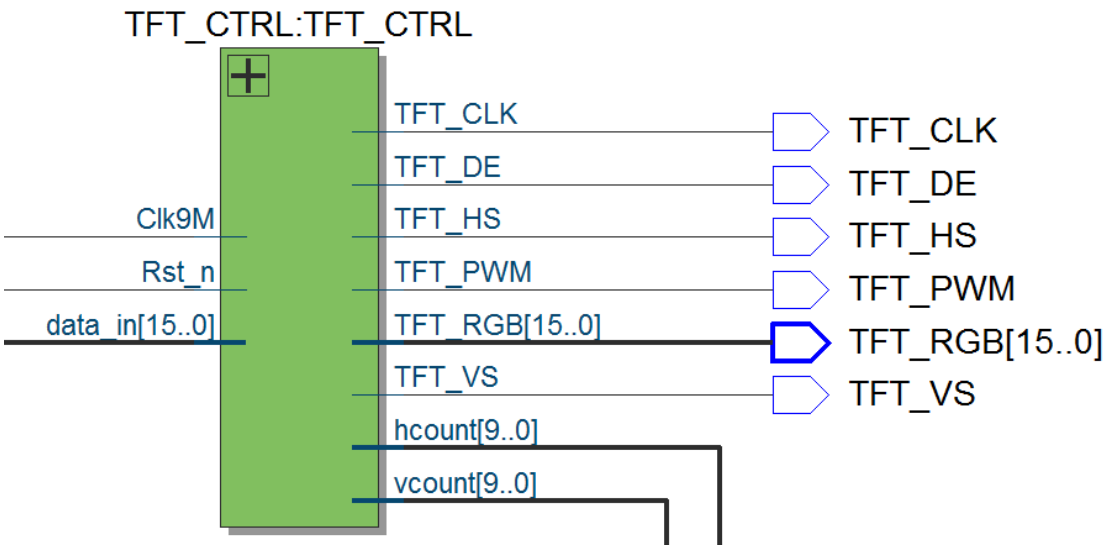
```
    else if(hcount_ov)
        hcount_r<=10'd0;
    else
        hcount_r<=hcount_r+10'd1;

    assign hcount_ov=(hcount_r==hpixel_end);

    //场扫描
    always@(posedge Clk9M or negedge Rst_n)
    if(!Rst_n)
        vcount_r<=10'd0;
    else if(hcount_ov) begin
        if(vcount_ov)
            vcount_r<=10'd0;
        else
            vcount_r<=vcount_r+10'd1;
    end
    else
        vcount_r<=vcount_r;

    assign  vcount_ov=(vcount_r==vline_end);

    //数据、同步信号输出
    assign dat_act=((hcount_r>=hdat_begin)&&(hcount_r<hdat_end))
                  &&((vcount_r>=vdat_begin)&&(vcount_r<vdat_end));

    assign TFT_HS=(hcount_r>TFT_HS_end);
    assign TFT_VS=(vcount_r>TFT_VS_end);
    assign TFT_RGB=(dat_act)?data_in:16'h0000;

endmodule
```

相较于之前的 VGA 控制器设计，可以看到，代码方面除了一些时序参数有所改变，其他没有任何变化，只是新增了 3 个输出信号：

```
    TFT_CLK,        //TFT 像素时钟
    TFT_DE,     //TFT 数据使能
    TFT_PWM     //TFT 背光控制
```

而 TFT_DE 直接与 dat_act 信号相连，TFT_PWM 则直接与复位输入相连，TFT_CLK 直接与驱动器工作时钟也就是像素时钟相连。并未对整个控制器设计产生较大改变。

设计完成后，在 Quartus II15.1 中综合出来的电路符号如下所示：



每个端口的功能如下表所示：

端口名	端口功能
Clk9M	TFT 像素时钟，9MHz
Rst_n	系统复位，低电平复位
data_in[15:0]	待显示数据输入端口
TFT_HS	TFT 接口行同步信号
TFT_VS	TFT 接口场同步信号
TFT_RGB[15:0]	TFT 三元色数据输出
hcount[9:0]	图像区行扫描地址
vcount[9:0]	图像区场扫描地址

TFT 控制器仿真验证

本小节对设计的 TFT 控制器进行仿真验证，通过仿真查看行场同步信号是否满足设计需求。

Testbench 设计

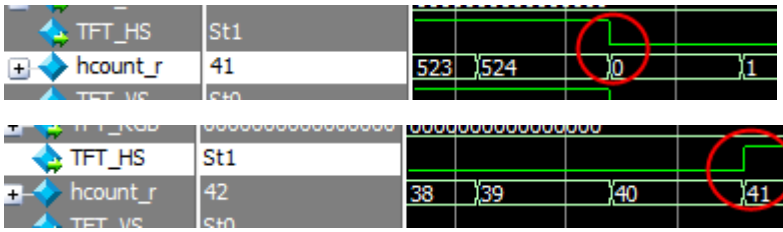
Testbench 的设计思路非常简单，只需要产生一个 25MHz 的时钟信号，然后在 data_in 端口上给一个固定的数据编码，为了与消隐时候的强制输出全 0 相区分，因此只需要是 data_in 上的数据不为 0 即可。testbench 内容如下所示：

```
`timescale 1ns/1ns

`define clk_period 120

module TFT_CTRL_tb;
    //-----------------模块输入端口----------------
    reg  Clk9M;          //系统输入时钟 9MHZ
    reg  Rst_n;
    reg  [7:0]data_in;     //待显示数据

    //-----------------模块输出端口----------------
    wire [9:0]hcount;
    wire [9:0]vcount;
    wire [7:0]TFT_RGB;  //TFT 数据输出
    wire TFT_HS;          //TFT 行同步信号
    wire TFT_VS;          //TFT 场同步信号

    reg [11:0]V_cnt = 0;//扫描行数统计计数器

    TFT_CTRL TFT_CTRL(
        .Clk9M(Clk9M),  //系统输入时钟 9MHZ
        .Rst_n(Rst_n),
        .data_in(data_in),  //待显示数据
        .hcount(hcount),        //TFT 行扫描计数器
        .vcount(vcount),        //TFT 场扫描计数器
        .TFT_RGB(TFT_RGB),  //TFT 数据输出
        .TFT_HS(TFT_HS),        //TFT 行同步信号
        .TFT_VS(TFT_VS)     //TFT 场同步信号
    );

    initial Clk9M = 0;
    always #(`clk_period/2) Clk9M = ~Clk9M;

    initial begin
        Rst_n = 0;
        data_in = 8'd0;
        #(`clk_period *20 +1);
        Rst_n = 1;
        data_in = 8'hff;
    end

    initial begin
        wait(V_cnt == 3);//等待扫描 2 帧后结束仿真
        $stop;
```

```
    end

    always @(posedge TFT_VS)//统计总扫描帧数
        V_cnt <= V_cnt + 1'b1;

endmodule
```

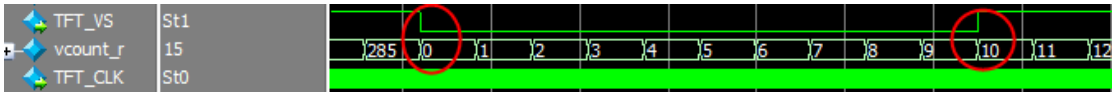
仿真结果分析

TFT_HS 信号：



由图可见，TFT_HS 在 0~41 这一行扫描段内为低电平，即行同步头，其他时间为高电平，行扫描一次，行扫描计数器计数最大值为 524，即刚好 525 个像素时钟周期，与设计一致，因此可知行扫描信号满足时序设计要求。

TFT_VS 信号：



由图可见，VGA_VS 信号在 0~10 这一段场扫描时间内为低电平，即场同步头，其他时间为高电平。场扫描一次，场扫描计数器计数最大值为 285，即刚好 286 个行扫描周期，与设计一致，满足 480*272 分辨率时序标准，因此可知场扫描信号满速时序设计要求。

其他信号本文不再进行详细分析比对，在进行板级调试中，如果发现显示效果不对，则可根据实际显示效果，判断错误位置，如行同步信号错误、场同步信号错误等。

TFT 控制器板级验证

在上一节，我们简述了 TFT 控制器的设计思路并给出了具体的 TFT 控制器设计过程，同时通过仿真验证了设计的合理性。本节，我们将对该 TFT 控制器进行板级验证，通过板级验证来进一步确定我们设计的正确性。

板级验证需求

TFT 的板级验证，主要验证以下三个方面：

1、能够正确的全屏点亮屏幕，显示稳定

2、能否正确的显示颜色，即按照需求制定需要显示的颜色

3、能否正确的定位坐标，即实现在指定的位置显示对应的数据

板级验证电路设计

为此，我们设计一个测试工程，该工程中我们测试上述提到的 8 种颜色，通过颜色的位置，不但能确定是否能够正确输出指定颜色的图像，还能间接确定是否能够精确指定像素位置。

因此，我们对屏幕进行划分，将屏幕划分成 4 行 2 列总共八个像素阵列，每个阵列分别显示一种颜色。据此，我们可以首先定义每种颜色的具体数据编码，然后再定义每个像素阵列的基本显示颜色，这里首先使用 localparam 定义每种颜色的具体数据编码：

//定义颜色编码

```
localparam
    BLACK   = 16'h0000, //黑色
    BLUE    = 16'h001F, //蓝色
    RED     = 16'hF800, //红色
    PURPPLE = 16'hF81F, //紫色
    GREEN   = 16'h07E0, //绿色
    CYAN    = 16'h07FF, //青色
    YELLOW  = 16'hFFE0, //黄色
    WHITE   = 16'hFFFF; //白色
```

//定义每个像素块的默认显示颜色值

```
localparam
    R0_C0 = BLACK,  //第 0 行 0 列像素块
    R0_C1 = BLUE,   //第 0 行 1 列像素块
    R1_C0 = RED,    //第 1 行 0 列像素块
    R1_C1 = PURPPLE,//第 1 行 1 列像素块
    R2_C0 = GREEN,  //第 2 行 0 列像素块
    R2_C1 = CYAN,   //第 2 行 1 列像素块
    R3_C0 = YELLOW, //第 3 行 0 列像素块
    R3_C1 = WHITE;  //第 3 行 1 列像素块
```

紧接着，我们需要知道 TFT 当前扫描的位置是在哪一个位置区间，换一种说法，我们需要通过 TFT 当前的扫描位置得到当前扫描的是哪一个像素阵列，然后给待显示数据赋予对应的颜色值即可。这里我们先定义每个像素块处于扫描中的条件。

1、产生每一列的处于扫描状态标志信号，屏幕每行总共 480 个像素点，我们将屏幕划分成了 2 列，因此

a) 当行扫描范围在 0~239 这一段像素内时，第 0 列处于活跃阶段；

b) 当扫描范围在 240~479 这一段像素内时，第 1 列处于活跃阶段。

因此可得：

```
wire C0_act = hcount >= 0 && hcount < 240; //正在扫描第 0 列
wire C1_act = hcount >= 240 && hcount < 480;//正在扫描第 1 列
```

2、产生每一行的处于扫描状态标志信号，屏幕每列总共 272 个像素点，我们将屏幕划分成了 4 列，因此

a) 当行扫描范围在 0~67 这一段像素内时，第 0 行处于活跃阶段；

b) 当行扫描范围在 68~135 这一段像素内时，第 1 行处于活跃阶段；

c) 当行扫描范围在 136~203 这一段像素内时，第 2 行处于活跃阶段；

d) 当行扫描范围在 204~272 这一段像素内时，第 3 行处于活跃阶段，

因此可得：

```
wire R0_act = vcount >= 0 && vcount < 68;   //正在扫描第 0 行
wire R1_act = vcount >= 68 && vcount < 136;//正在扫描第 1 行
wire R2_act = vcount >= 136 && vcount < 204;//正在扫描第 2 行
wire R3_act = vcount >= 204 && vcount < 272;//正在扫描第 3 行
```

3、产生扫描每一个像素块的标志信号：

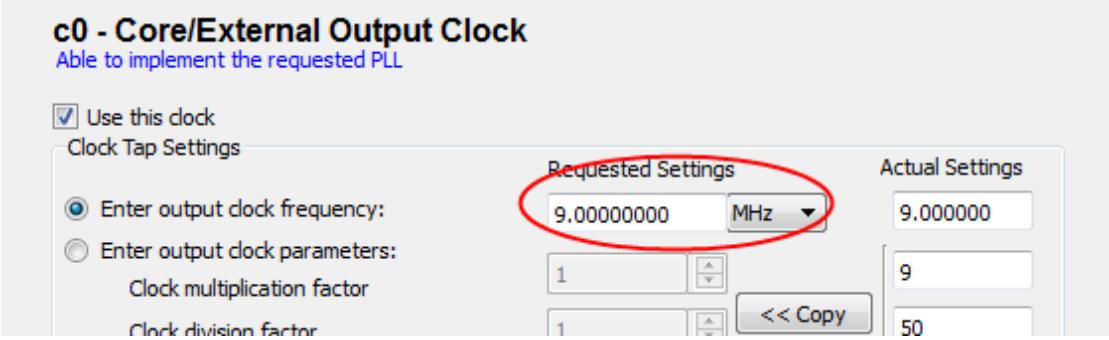
```
wire R0_C0_act = R0_act & C0_act;   //第 0 行 0 列像素块被扫描中
wire R0_C1_act = R0_act & C1_act;   //第 0 行 1 列像素块被扫描中
wire R1_C0_act = R1_act & C0_act;   //第 1 行 0 列像素块被扫描中
wire R1_C1_act = R1_act & C1_act;   //第 1 行 1 列像素块被扫描中
wire R2_C0_act = R2_act & C0_act;   //第 2 行 0 列像素块被扫描中
wire R2_C1_act = R2_act & C1_act;   //第 2 行 1 列像素块被扫描中
wire R3_C0_act = R3_act & C0_act;   //第 3 行 0 列像素块被扫描中
wire R3_C1_act = R3_act & C1_act;   //第 3 行 1 列像素块被扫描中
```

然后，我们就可以根据当前被扫描的像素块范围来确定需要给 TFT 输出什么颜色，这里采用一个多路器即可实现：

```
always@(*)
    case({R3_C1_act,R3_C0_act,R2_C1_act,R2_C0_act,
          R1_C1_act,R1_C0_act,R0_C1_act,R0_C0_act})
        8'b0000_0001:disp_data = R0_C0;
        8'b0000_0010:disp_data = R0_C1;
        8'b0000_0100:disp_data = R1_C0;
        8'b0000_1000:disp_data = R1_C1;
        8'b0001_0000:disp_data = R2_C0;
        8'b0010_0000:disp_data = R2_C1;
        8'b0100_0000:disp_data = R3_C0;
        8'b1000_0000:disp_data = R3_C1;
        default:disp_data = R0_C0;
    endcase
```

添加 PLL 时钟分频单元

通过以上步骤，我们就完成了简易 TFT 控制器测试电路的主要电路设计。在前面我们曾经提到，TFT 控制器的像素时钟为 9MHz，而我们芯航线 FPGA 开发板设计的是 50MHz 的晶振，因此需要使用锁相环对时钟进行分频得到 9MHz 的时钟，以供 TFT 控制器使用。具体 PLL 配置请参考《芯航线 FPGA 数字系统设计教程+实例解析》“FPGA 设计思想与验证方法视频教程实验精讲手册”部分的“十六、 PLL 锁相环介绍与简单应用”小节。



完整的测试电路代码

实现完整的测试电路代码如下所示：

```
module TFT_CTRL_test(
    Clk,    //50MHZ 时钟
    Rst_n,
    TFT_RGB,//TFT 数据输出
    TFT_HS, //TFT 行同步信号
    TFT_VS, //TFT 场同步信号
    TFT_CLK,
    TFT_DE,
    TFT_PWM
);

    input Clk;
    input Rst_n;
    output [15:0]TFT_RGB;
    output TFT_HS;
    output TFT_VS;
    output TFT_CLK;
    output TFT_DE;
    output TFT_PWM;

    reg [15:0]disp_data;
    wire [9:0]hcount;
    wire [9:0]vcount;
    wire Clk9M;

    TFT_test_pll TFT_test_pll(
        .inclk0(Clk),
        .c0(Clk9M)
    );
```

```
    TFT_CTRL TFT_CTRL(
        .Clk9M(Clk9M),  //系统输入时钟 9MHZ
        .Rst_n(Rst_n),
        .data_in(disp_data),    //待显示数据
        .hcount(hcount),        //TFT 行扫描计数器
        .vcount(vcount),        //TFT 场扫描计数器
        .TFT_RGB(TFT_RGB),  //TFT 数据输出
        .TFT_HS(TFT_HS),        //TFT 行同步信号
        .TFT_VS(TFT_VS),        //TFT 场同步信号
        .TFT_CLK(TFT_CLK),
        .TFT_DE(TFT_DE),
        .TFT_PWM(TFT_PWM)
    );


//定义颜色编码
localparam
    BLACK      = 16'h0000, //黑色
    BLUE       = 16'h001F, //蓝色
    RED     = 16'hF800, //红色
    PURPPLE = 16'hF81F, //紫色
    GREEN      = 16'h07E0, //绿色
    CYAN       = 16'h07FF, //青色
    YELLOW  = 16'hFFE0, //黄色
    WHITE      = 16'hFFFF; //白色

//定义每个像素块的默认显示颜色值
localparam
    R0_C0 = BLACK,  //第 0 行 0 列像素块
    R0_C1 = BLUE,   //第 0 行 1 列像素块
    R1_C0 = RED,    //第 1 行 0 列像素块
    R1_C1 = PURPPLE,//第 1 行 1 列像素块
    R2_C0 = GREEN,  //第 2 行 0 列像素块
    R2_C1 = CYAN,   //第 2 行 1 列像素块
    R3_C0 = YELLOW, //第 3 行 0 列像素块
    R3_C1 = WHITE;  //第 3 行 1 列像素块

    wire R0_act = vcount >= 0 && vcount < 68;   //正在扫描第 0 行
    wire R1_act = vcount >= 68 && vcount < 136;//正在扫描第 1 行
    wire R2_act = vcount >= 136 && vcount < 204;//正在扫描第 2 行
    wire R3_act = vcount >= 204 && vcount < 272;//正在扫描第 3 行

    wire C0_act = hcount >= 0 && hcount < 240; //正在扫描第 0 列
```

```
    wire C1_act = hcount >= 240 && hcount < 480;//正在扫描第 1 列

    wire R0_C0_act = R0_act & C0_act;   //第 0 行 0 列像素块处于被扫描中
    wire R0_C1_act = R0_act & C1_act;   //第 0 行 1 列像素块处于被扫描中
    wire R1_C0_act = R1_act & C0_act;   //第 1 行 0 列像素块处于被扫描中
    wire R1_C1_act = R1_act & C1_act;   //第 1 行 1 列像素块处于被扫描中
    wire R2_C0_act = R2_act & C0_act;   //第 2 行 0 列像素块处于被扫描中
    wire R2_C1_act = R2_act & C1_act;   //第 2 行 1 列像素块处于被扫描中
    wire R3_C0_act = R3_act & C0_act;   //第 3 行 0 列像素块处于被扫描中
    wire R3_C1_act = R3_act & C1_act;   //第 3 行 1 列像素块处于被扫描中

    always@(*)
        case({R3_C1_act,R3_C0_act,R2_C1_act,R2_C0_act,
              R1_C1_act,R1_C0_act,R0_C1_act,R0_C0_act})
            8'b0000_0001:disp_data = R0_C0;
            8'b0000_0010:disp_data = R0_C1;
            8'b0000_0100:disp_data = R1_C0;
            8'b0000_1000:disp_data = R1_C1;
            8'b0001_0000:disp_data = R2_C0;
            8'b0010_0000:disp_data = R2_C1;
            8'b0100_0000:disp_data = R3_C0;
            8'b1000_0000:disp_data = R3_C1;
            default:disp_data = R0_C0;
        endcase

endmodule
```

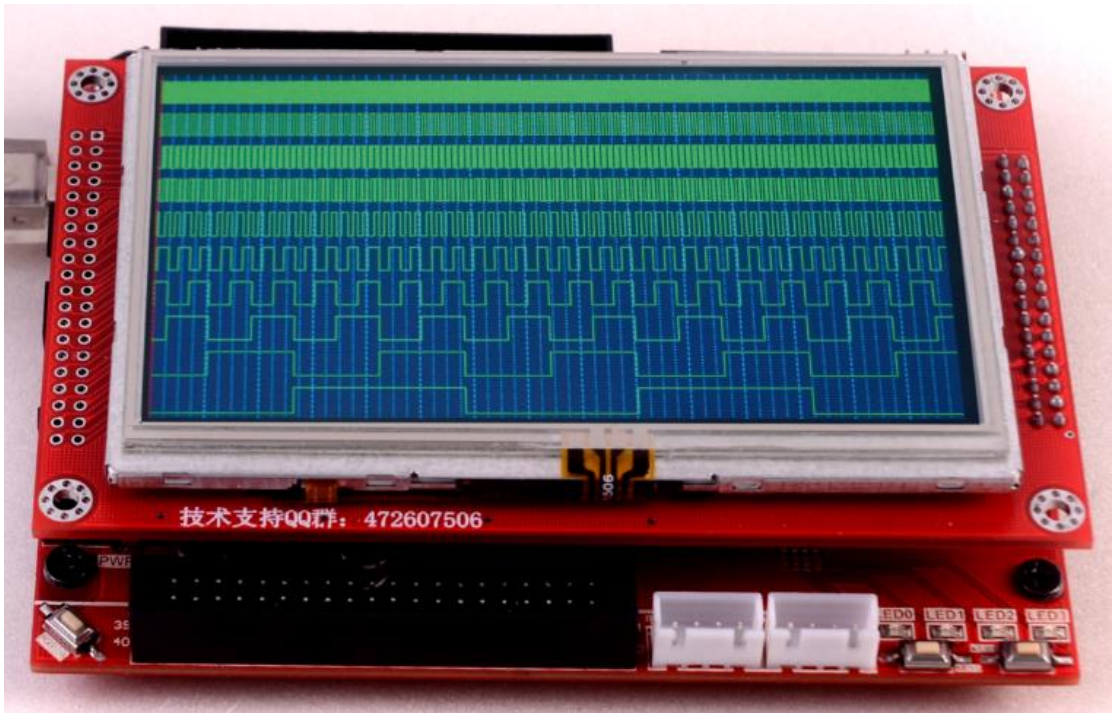
板级验证

引脚分配，TFT 控制器测试工程引脚分配表如下所示：

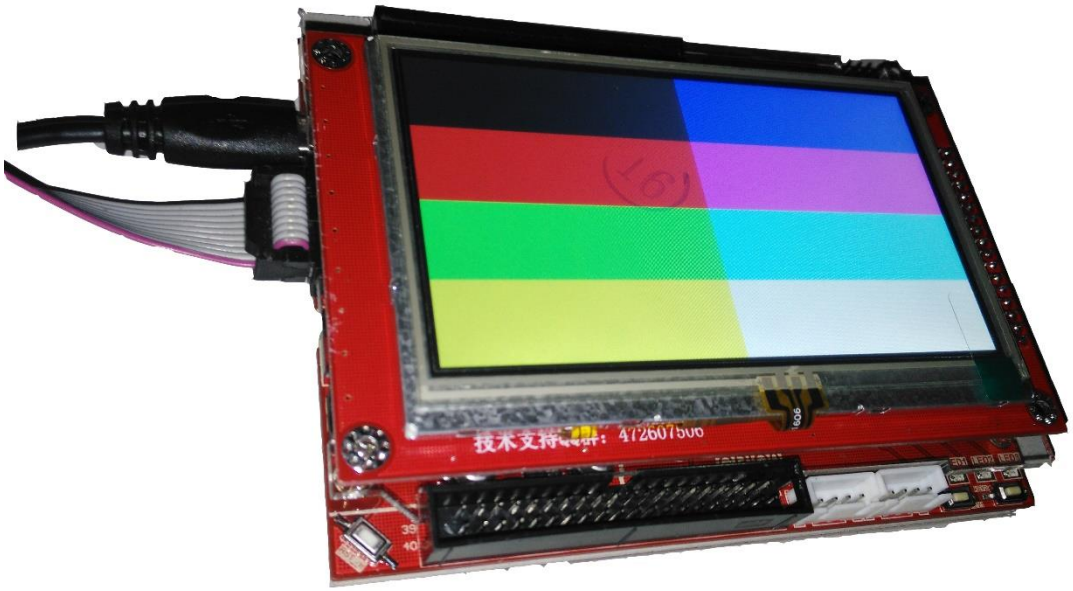
TFT 信号名	对应 FPGA 引脚名称
Clk	PIN_E1
Rst_n	PIN_M1
TFT_CLK	PIN_D14
TFT_DE	PIN_F10
TFT_HS	PIN_F13
TFT_PWM	PIN_D12
TFT_RGB[15]	PIN_L13
TFT_RGB[14]	PIN_E10
TFT_RGB[13]	PIN_C14
TFT_RGB[12]	PIN_M12

TFT_RGB[11]	PIN_N14
TFT_RGB[10]	PIN_J11
TFT_RGB[9]	PIN_K11
TFT_RGB[8]	PIN_J14
TFT_RGB[7]	PIN_K12
TFT_RGB[6]	PIN_L12
TFT_RGB[5]	PIN_L14
TFT_RGB[4]	PIN_G11
TFT_RGB[3]	PIN_K8
TFT_RGB[2]	PIN_K9
TFT_RGB[1]	PIN_L10
TFT_RGB[0]	PIN_K10
TFT_VS	PIN_F14

芯航线 FPGA 学习套件主板与“4.3 寸触摸显示组件”的连接如下所示：



最终测试效果如下图所示：



通过照片可知，TFT 控制器设计能够稳定正确的刷新 TFT 显示屏并控制正确的显示位置，因此设计无误。

后续，我们就可以使用该控制器再结合一定的图像信号产生电路实现更多更负责的显示系统设计。当然，也可能根据具体的使用环境，再对本控制器进行设计微调。

如有更多问题，欢迎加入芯航线 FPGA 技术支持群交流学习：472607506

小梅哥

芯航线电子工作室

关于学习资料，小梅哥系列所有能够开放的资料和更新（包括视频教程，程序代码，教程文档，工具软件，开发板资料）都会发布在我的云分享。（记得订阅）链接：http://yun.baidu.com/share/home?uk=402885837&view=share#category/type=0