

32 基于 SiI9022 的 IC_HDMI/DVI 显示

工程源码	---02_设计实例 ---ch32_acz7015_colour_bar_ic_hdmi
相关视频课程	
	为了方便区分，在工程命名时，我们使用 ic_hdmi 指代使用 IC 芯片实现 HDMI 显示的设计；用 io_hdmi 指代通过逻辑设计，使用 IO 接口实现 HDMI 显示的设计。

章节导读

随着网络带宽、存储器容量、处理器解码能力的不断发展，与我们息息相关的图像显示领域也出现了非常大的进步。以显示器显示原理来说，从最早期的 CRT 阴极射线管扫描成像开始，到等离子电视，再到现在随处可见的液晶电视、以及现在高端领域里面使用的 OLED 显示器，整个显示器领域从最早期的黑白显示效果慢慢进化到现在的 OLED 自发光真彩色显示，视觉效果大大提升了。再从显示器的物理分辨率来说，早期的 CRT 显示器，其最大分辨率为 800*600 像素，而现在的液晶显示器，分辨率已经一路从 720p 发展到 1080p、2K、4K 以及最新的 8K 分辨率，显示效果更加细腻，色彩还原效果也不断提升。

随着显示技术的发展，与显示技术密切相关的数据传输方式和能力也在不断的提升。传输方式从最早期的 VGA 模拟信号传输方式发展到 DVI 接口、HDMI 接口、DisplayPort 接口等，传输的数据质量，数据容量也都有了大的提升。传输方式的变化主要与成像原理和数据速率相关。

ACZ7015 开发板在硬件上支持 HDMI 接口，开发板上设计有一颗 IC 芯片 SiI9022，用于 RGB 到 HDMI 的转换。本节将对 SiI9022 芯片作简单的介绍，并基于该芯片实现 DVI 接口的显示器彩条显示实验。

32.1HDMI 芯片 SiI9022

为了简化用户设计，ACZ7015 开发板上使用了一颗专用的 RGB 转 HDMI 芯片 SiI9022ACNU。该芯片支持 IIC 接口，使用时，用户只需要通过 IIC 总线配置芯片工作模式，并提供显示所需的 RGB 图像数据和时序信号，芯片便会自动对信号进行转换和编码，最终以符合 HDMI/DVI 接口的格式输出。

如此，用户只需了解 IIC 总线协议，便能完成 HDMI 的显示。关于该 RGB 转 HDMI 专用芯片，芯片手册中给出了其型号编码的具体含义，如图 32-1 所示：

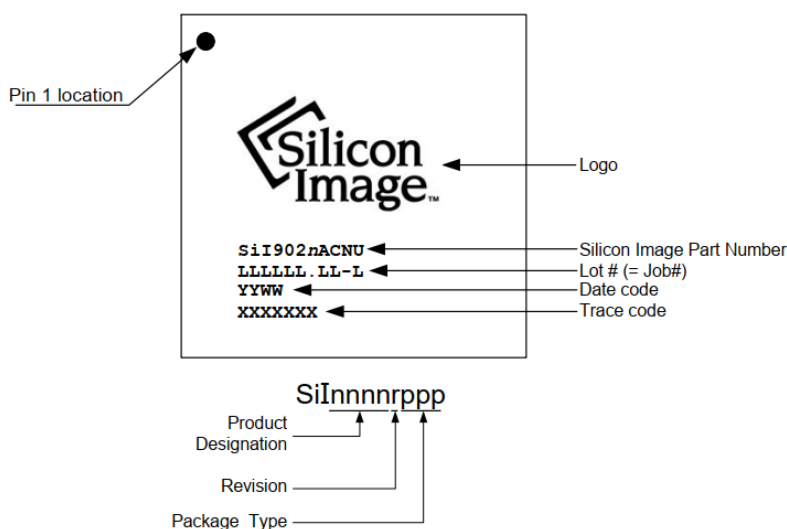


图 32-1 SiI9022ACNU 标记示意图

通过编码我们可以知道芯片的型号、版本以及封装，官方手册 SiI9022 中给出了不同编码芯片的相关信息，如图 32-2 所示：

Part Numbers	Package Type	Pixel Clock Range	Security	Temperature Grade
SiI9022ARBT	81-ball 4 × 4 mm VFBGA	25 MHz ~165 MHz	—	Extended (–20 °C to +85 °C)
SiI9022AYBT	49-ball 4 × 4 mm VFBGA	25 MHz ~165 MHz	—	Extended (–20 °C to +85 °C)
SiI9022ACNU	72-pin 10 × 10 mm QFN	25 MHz ~165 MHz	—	Extended (–20 °C to +85 °C)
SiI9024ARBT	81-ball 4 × 4 mm VFBGA	25 MHz ~165 MHz	HDCP	Extended (–20 °C to +85 °C)
SiI9024AYBT	49-ball 4 × 4 mm VFBGA	25 MHz ~165 MHz	HDCP	Extended (–20 °C to +85 °C)
SiI9024ACNU	72-pin 10 × 10 mm QFN	25 MHz ~165 MHz	HDCP	Extended (–20 °C to +85 °C)

图 32-2 芯片信息

从图中可以看到，ACZ7015 开发板上使用的 SiI9022ACNU 芯片为 72pin 引脚的 10x10mmQFN 封装，与 SiI9022 系列其他编码的芯片，仅在引脚和封装工艺上存在差别，而在编程方面区别不大，为了方便，后文中统一以 SiI9022 指代开发板上的 SiI9022ACNU 芯片。

SiI9022 芯片的功能框图如图 32-3：

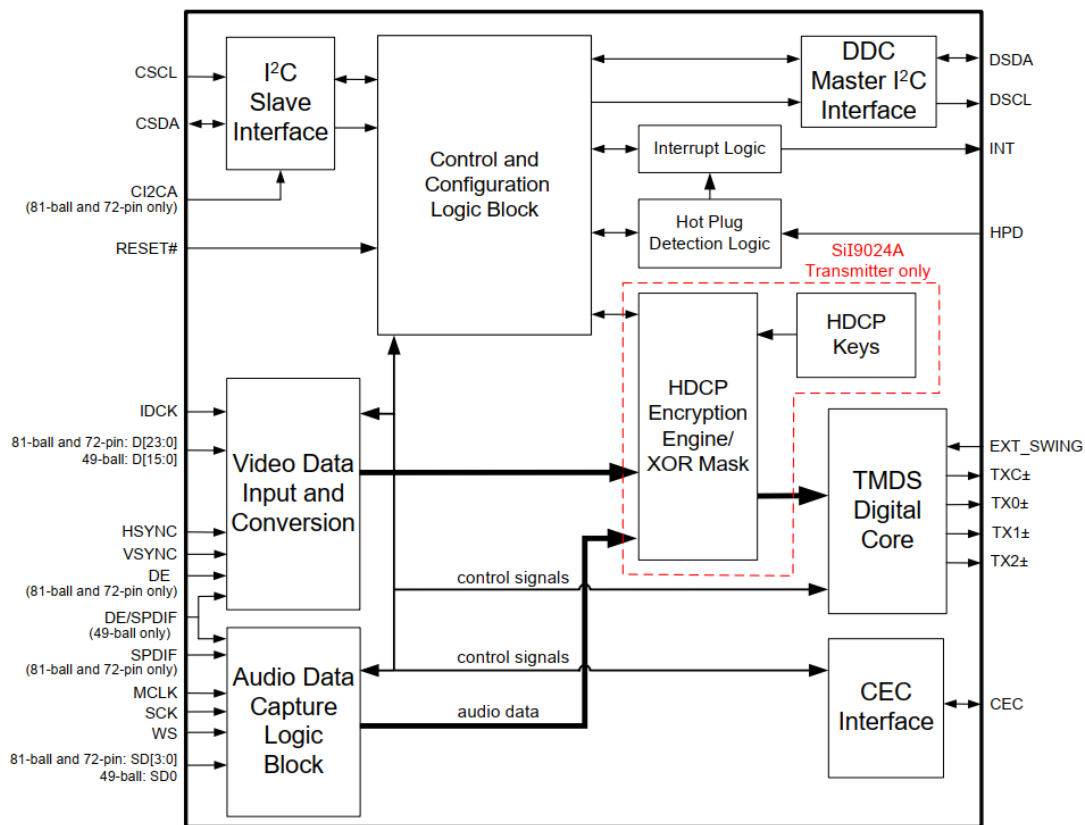


图 32-3 功能框图

其中红色虚线部分为 SiI9024A 独有的加密/解密模块，SiI9022 仅支持虚线外的各个模块功能，其各自的功能如下：

- Video Data Input and Conversion: 视频数据输入转换模块
- Audio Data Capture Logic Block: 音频数据输入模块
- TMDs Digital Core: 实现数据的 TMDs 编码
- I²C Slave Interface: I²C 从接口，用于写入相关寄存器修改配置；
- Control and Configuration Logic Block: 控制和配置逻辑块，管理着所有用于配置和管理发送器的寄存器；
- DDC Master I²C Interface: 用于获取显示器的相关参数，诸如可接收行场频范围、标准显示模式及其参数、所支持的 DDC 标准类别、EDID 的版本信息等等；
- CEC Interface: 用于为通过 HDMI 连接的视听设备提供高级控制功能。
- Interrupt Logic: 中断逻辑，产生的中断信号会中断主机处理器
- Hot Plug Detection Logic: 热插拔检测逻辑，用于检测 HDMI 是否连接。

用户的配置信号会通过 IIC 总线，传输给 IIC 从接口，其中的控制和配置逻辑会根据配置信号设置相关寄存器。基于这些配置，用户通过 Video Data Input

店铺: <https://xiaomeige.taobao.com> 官方网站: www.corecourse.cn
 技术博客: <http://www.cnblogs.com/xiaomeige/> 技术群组:

and Conversion 模块输入的图像数据，会经由 TMDS Digital Core 编码转换成 HDMI 接口所需的 TMDS 编码格式并输出。

SiI9022 在 ACZ7015 开发板上对应的电路如图 32-4 所示:

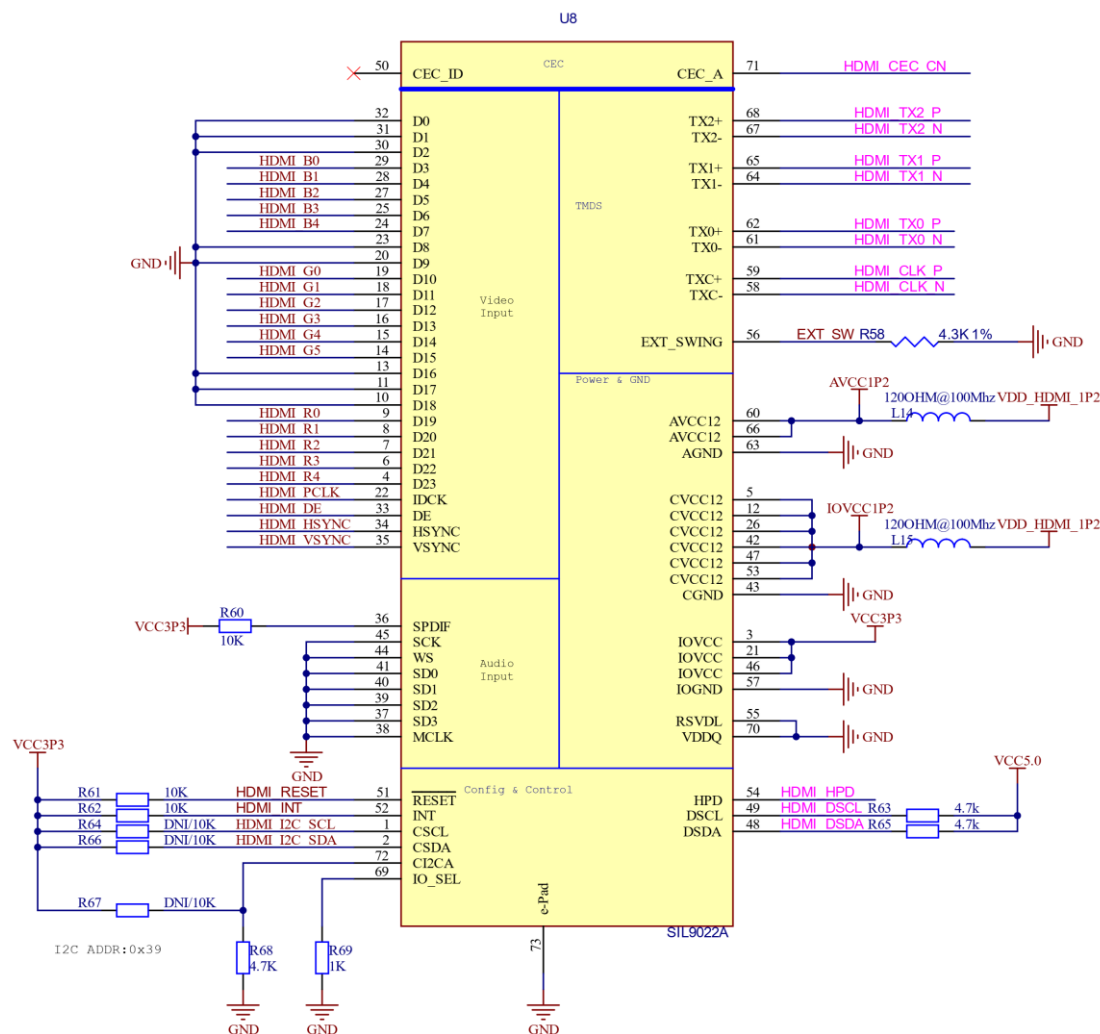


图 32-4 Si19022 电路图

从图 32-4 中可以看到，ACZ7015 开发板在进行硬件设计时将 SiI9022 的音频输入引脚（图中 Audio Input 部分）接地，因此 ACZ7015 开发板只支持图像视频数据的 HDMI 传输。

同时，SiI9022 的 R、G、B 颜色分量引脚接口各有 8 位，但是低位的数据引脚在硬件上物理接地，因此这些引脚的输入值固定为 0，也就是我们常说的低位补零。剩余的 RGB565 引脚在硬件上与 TFT 屏的 RGB565 数据引脚共享，如图 32-5 所示：

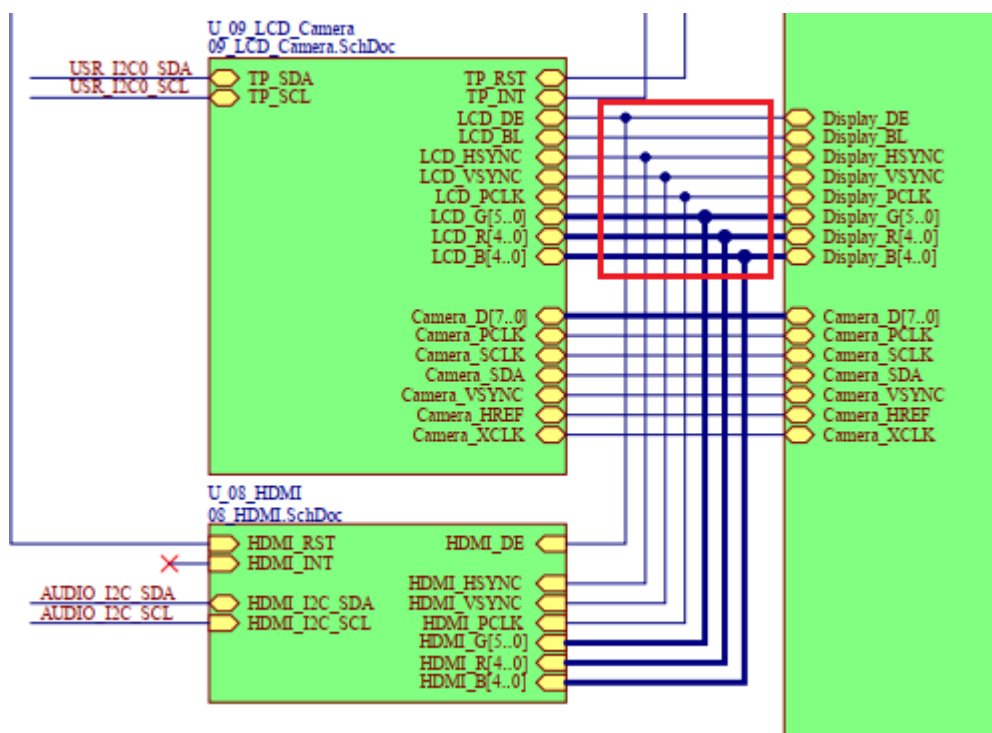


图 32-5 SiI9022 数据与时序信号引脚电路

可以看到，除了数据引脚外，二者共享的还有时序引脚。因此，在使用 ACZ7015 开发板进行 HDMI 设计时，需要分配 TFT 数据和时序信号引脚，用以给 SiI9022 芯片传输显示相关信号。

除了这些之外，我们还可以看到，用于配置 SiI9022 的 I2C 接口，其名字前缀为 AUDIO。这是因为在一条 I2C 总线能够挂载多个从设备，在设计 ACZ7015 开发板电路时，我们将 SiI9022 与音频编解码模块 WM8960 一同挂载在了同一条 I2C 总线上。如图 32-6 所示：

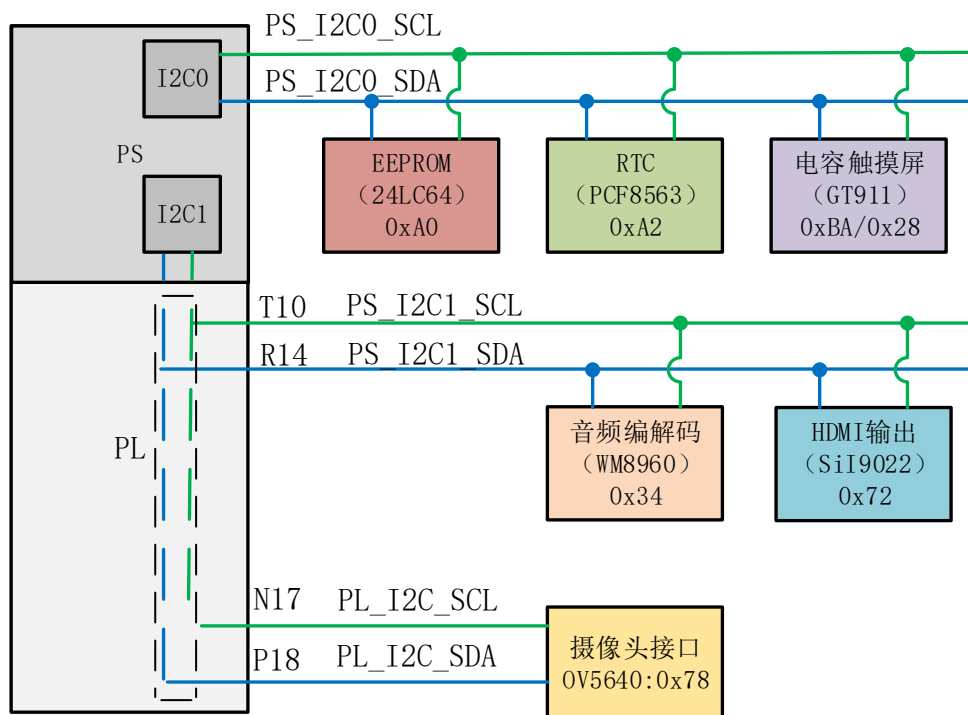


图 32-6 ACZ7015 开发板 IIC 设备分布

从图中我们可以知道，SiI9022 的器件地址为 0x72，基于此，我们便能通过 I2C 总线配置 SiI9022，令 SiI9022 对输入的数据和时序信号处理转换，输出 HDMI 所需格式信号。至于如何配置并初始化 SiI9022，便是我们接下来要讲的内容。

32.1.1 SiI9022 初始化配置表

为了方便用户操作，我们制作并提供了 SiI9022 的初始化表，用户只需使用 IIC 协议按照配置表中的内容，配置对应寄存器即可。配置表内容如下：

```
`timescale 1ns / 1ps
module SiI9022_init_table
#(parameter DATA_WIDTH=16, parameter ADDR_WIDTH=8)
(
    input [(ADDR_WIDTH-1):0] addr,
    input clk,
    output reg [(DATA_WIDTH-1):0] q
);

reg [DATA_WIDTH-1:0] rom[2**ADDR_WIDTH-1:0];

initial begin
    rom[00] = 16'hc7_00;
    rom[01] = 16'hBC_01;
```

```
rom[02] = 16'hBD_00;
rom[03] = 16'hBE_01;
rom[04] = 16'h00_02;
rom[05] = 16'h01_3A;
rom[06] = 16'h02_70;
rom[07] = 16'h03_17;
rom[08] = 16'h04_98;
rom[09] = 16'h05_08;
rom[10] = 16'h06_65;
rom[11] = 16'h07_04;
rom[12] = 16'h08_60;
rom[13] = 16'h09_00; // ---RGB input
rom[14] = 16'h0A_00; // ---RGB output
rom[15] = 16'h0B_00; //Set AVI
rom[16] = 16'h0C_00;
rom[17] = 16'h0D_00;
rom[18] = 16'h0E_00;
rom[19] = 16'h0F_00;
rom[20] = 16'h10_00;
rom[21] = 16'h11_00;
rom[22] = 16'h12_00;
rom[23] = 16'h13_00;
rom[24] = 16'h14_00;
rom[25] = 16'h15_00;
rom[26] = 16'h16_00;
rom[27] = 16'h17_00;
rom[28] = 16'h18_00;
// ---enable color convert, must do it here
rom[29] = 16'h19_00;
rom[30] = 16'h1E_00;
rom[31] = 16'h60_04; // ---embeded sync select
rom[32] = 16'h62_00; // ---HBIT_2_HSYNC L
// ---embede enable and HBIT_2_HSYNC M //bit6=1 for DE
rom[33] = 16'h63_00;
rom[34] = 16'h64_00; // ---FIELD2_OFST L
rom[35] = 16'h65_04; // ---FIELD2_OFST M
rom[36] = 16'h66_00; // ---HWIDTH L
rom[37] = 16'h67_00; // ---HWIDTH M
rom[38] = 16'h68_00; // ---VBIT_2_VSYNC
rom[39] = 16'h69_00; // ---VWIDTH
rom[40] = 16'hbf_c0; // ---VWIDTH
rom[41] = 16'h1A_00; //Enable TMDS

end

always @ (posedge clk)
begin
    q <= rom[addr];
end
```



```
end
endmodule
```

配置表中每一个 rom 的高 8 位用来存放寄存器地址，低 8 位用来存放对应寄存器的配置值。该配置表会根据用户输入的 ROM 地址，在对应时钟上升沿输出对应 ROM 表中的值。这里我们简单介绍下其中几个常用的寄存器的部分位，如表 32-1 所示：

表 32-1 SiI9022 部分寄存器一览

寄存器	位	默认值	功能描述
0x00	7:0	0x00	像素时钟低八位，像素时钟/10000
0x01	7:0	00000000	像素时钟高八位
0x02	7:0	00000000	垂直频率低八位，单位为 Hz
0x03	7:0	00000000	垂直频率高八位
0x04	7:0	00000000	行像素低八位
0x05	7:0	00000000	行像素高八位
0x06	7:0	00000000	图像行数低八位
0x07	7:0	00000000	图像行数高八位
0x09	7:6		输入颜色深度 00: 8bit 01: 保留 10: 10/12 位无抖动，适用于 4:2:2 模式 11: 10/12 位抖动，适用于 4:2:2 模式
	1:0		输入色彩空间 00: RGB 01: YCbCr 4:4:4 10: YCbCr 4:2:2 11: 黑色模式
0x0A	1:0		输出格式 00: RGB 01: YCbCr 4:4:4 10: YCbCr 4:2:2 11: RGB（等同于 00）
0x60	7		同步方式 0: 外同步 1: 内同步
0x1A	4		TMDS 输出控制 0: 工作 1: 掉电
	0		输出模式选择 0: DVI 模式 1: HDMI 模式

结合寄存器表我们可以知道，设计中将 SiI9022 的同步方式配置为了外同步（寄存器 0x60[7]=1），因此 SiI9022 将会使用从 TFT 电路中输入的相关时序信号

以及像素时钟。

如果配置模式设置为内同步，除了寄存器表中给出的 0x00~0x08 寄存器外，用户还需要设置其他一系列未列出的寄存器，将会使得设计更为麻烦。

除了以上配置外，配置表将 SiI9022 配置为 RGB888 输入输出格式，启用 TMDS 编码。同时，考虑到响应速度，配置表中默认的输出模式为 DVI，用户可以通过修改 0x1A[0] 的值为 1 切换成 HDMI 模式。

32.1.2 SiI9022 初始化模块

在有了 SiI9022 配置表后，接下来我们便可以开始设计 SiI9022 初始化模块。初始化 SiI9022 需要读取配置表中的内容，然后使用 I2C 总线对 SiI9022 芯片配置，因此本次设计需要复用前面章节中设计完成的 i2c_control 模块。

本次设计 SiI9022 初始化模块的简易结构框图如图 32-7 所示：

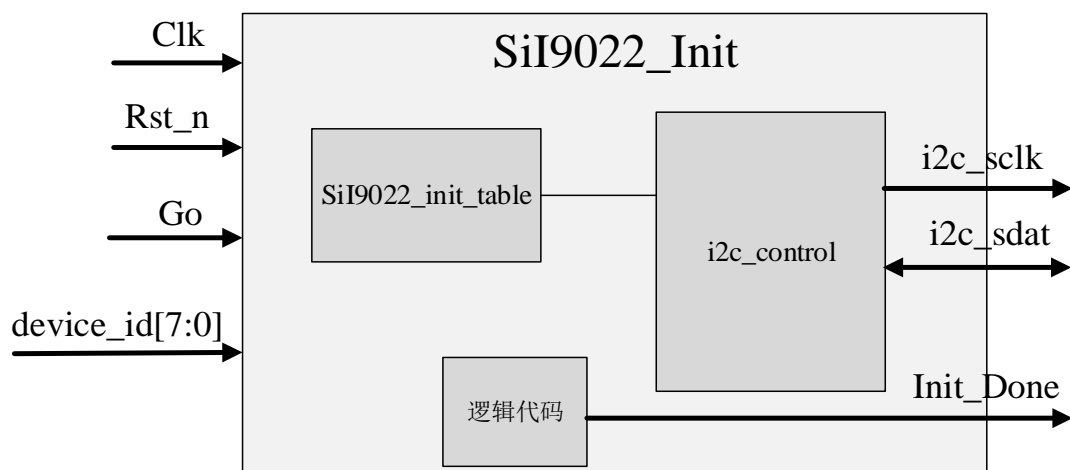


图 32-7 SiI9022 初始化模块结构框图

首先是对两个模块的例化和连接，由于对 SiI9022 的初始化，只需要进行写操作，因此 I2C 控制器的读请求信号我们可以直接赋值 0。代码如下：

```
SiI9022_init_table SiI9022_init_table(  
    .addr(cnt),  
    .clk(Clk),  
    .q(lut)  
);  
  
i2c_control i2c_control(  
    .Clk(Clk),  
    .Rst_n(Rst_n),  
    .wrreg_req(wrreg_req),  
    .rdreg_req(0),  
    .addr(addr),
```

```
.addr_mode(addr_mode),  
.wdata(wdata),  
.rdata(rdata),  
.device_id(device_id),  
.RW_Done(RW_Done),  
.ack(ack),  
.i2c_sclk(i2c_sclk),  
.i2c_sdat(i2c_sdat)  
);
```

模块图中，输入信号 Go 为脉冲信号，用于使能 SiI9022 的初始化。为了能够依次读取出 ROM 中的值，我们需要通过时序逻辑产生一个自加的 cnt 信号，作为 ROM 的地址，从配置表中读取数据。代码如下：

```
always@(posedge Clk or negedge Rst_n)  
if(!Rst_n)  
    cnt <= 0;  
else if(Go)  
    cnt <= 0;  
else if(cnt < lut_size)begin  
    if(RW_Done && (!ack))  
        cnt <= cnt + 1'b1;  
    else  
        cnt <= cnt;  
end  
else  
    cnt <= 0;
```

从配置表中读出的数据 lut 由 8 位的寄存器地址和 8 位的配置值组成，因此我们需要从中取出对应的数据，代码如下：

```
assign addr = lut[15:8];  
assign wdata = lut[7:0];
```

这些地址和配置数据会被送给 I2C 控制器，为了让控制器能够基于该配置值配置对应寄存器，我们需要产生写请求信号。代码如下：

```
always@(posedge Clk or negedge Rst_n)  
if(!Rst_n)begin  
    state <= 0;  
    wrreg_req <= 1'b0;  
end  
else if(cnt < lut_size)begin  
    case(state)  
        0:  
            if(Go)  
                state <= 1;  
            else  
                state <= 0;
```

```
1:
    begin
        wrreg_req <= 1'b1;
        state <= 2;
    end

2:
    begin
        wrreg_req <= 1'b0;
        if(RW_Done)
            state <= 1;
        else
            state <= 2;
        end
    end

default:state <= 0;
endcase
end
else
    state <= 0;
```

写请求信号会在初始化开始时拉高，随后立马被拉低，以产生一个高脉冲。当 I2C 控制器接收到高脉冲的写请求信号后，便开始对其中一个寄存器写配置。在配置完成后，I2C 控制器会产生 RW_Done 信号，说明一次传输完成。紧接着状态机会产生下一个高脉冲的写请求信号，准备传输下一轮数据。如此，在配置完所有寄存器后，状态机将会停止在状态 2，而在下一次 Go 信号脉冲到来前，写请求信号将一直为 0。

到这里，我们便完成了 SiI9022 初始化模块的设计，接下来，我们将通过一个具体的工程对设计模块进行板级测试，以验证设计的正确性。

32.2 基于 DVI 接口的显示器彩条显示实验

在设计完 SiI9022_Init 模块后，我们只需要以该模块为基础，为设计提供能够正常显示用的图像数据和时序信号，便能对模块进行验证。本次设计我们依然通过彩条显示实验为设计提供图像数据输入，整个系统的结构框图如图 32-8 所示：

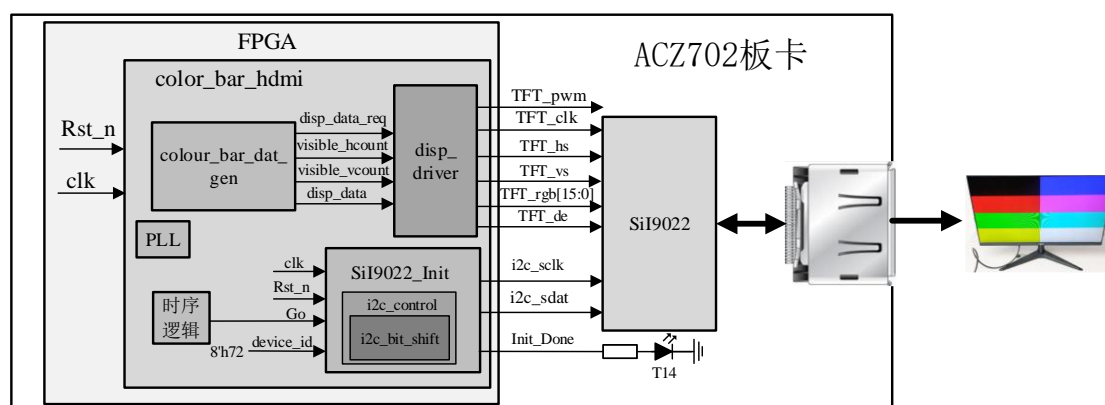


图 32-8 基于 DVI 接口的显示器彩条显示系统

考虑到 SiI9022 的初始化配置表中使用的 1920*1080 分辨率，整个设计都基于 1920*1080 分辨率设计。其中，PLL 负责产生时序所需的 148.5MHz 时钟。tft_ctrl_test 模块会产生 1920*1080 的彩条图像数据，随后通过 disp_driver 模块生成对应的时序信号和图像数据输出给 SiI9022。

为了避免 SiI9022_Init 模块一直对 SiI9022 芯片初始化，我们需要使用时序逻辑产生一个高脉冲的 Go 信号，完成对 SiI9022 芯片的单次初始化。同时，为了方便确认对 SiI9022 的初始化是否完成，设计中将 Init_Done 信号路由到 PL 侧的 LED 上，若 LED 亮起，则代表初始化成功，否则便是初始化失败。

考虑到本次设计中 SiI9022 的初始化配置表中使用的 1920*1080 分辨率，我们需要修改 tft_ctrl_test 模块，以使其产生对应分辨率的图像数据和时序信号。

32.2.1 修改 tft_ctrl_test 模块

(1) 修改 PLL 输出时钟

1920*1080 分辨率所需的像素时钟为 148.5Mhz，因此修改 PLL 的输出时钟为 148.5MHz，用于 disp_driver 模块产生相关时序信号，如图 32-9 所示：

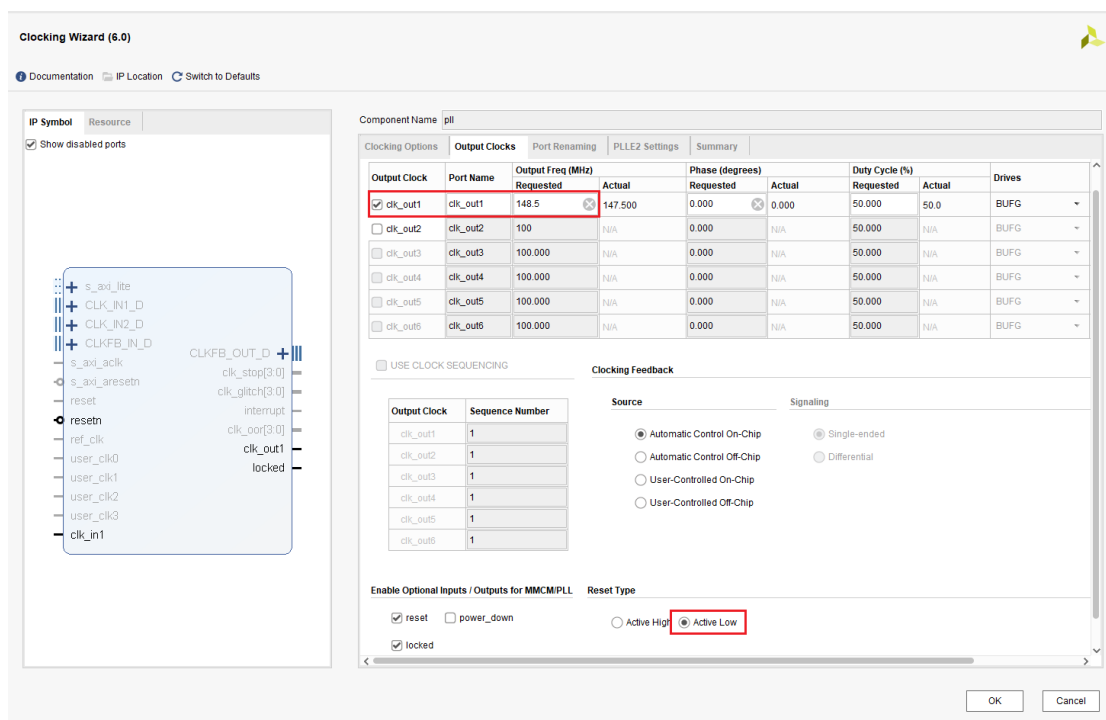


图 32-9 配置 PLL 输出时钟

(2) 修改 disp_parameter_cfg.v 文件

由于本次设计中显示用的分辨率变为 1920*1080，因此，我们需要修改 disp_parameter_cfg.v 文件中的 RGB 模式和分辨率参数配置，如下：

```
//使用 4.3 寸 480*272 分辨率显示屏
//`define HW_TFT43

//使用 5 寸 800*480 分辨率显示屏
//`define HW_TFT50

//使用 VGA 显示器，默认为 640*480 分辨率，24 位模式，其他分辨率或需 16 位模式
//可在代码 63 行至 75 行进行重配置
`define HW_VGA

//=====
//以下宏定义选择用于根据显示设备进行位模式和分辨率 2 个参数的设置
//=====
`ifndef HW_TFT43 //使用 4.3 寸 480*272 分辨率显示屏
    `define MODE_RGB565
    `define Resolution_480x272 1 //时钟为 9MHz
`elseif HW_TFT50 //使用 5 寸 800*480 分辨率显示屏
    `define MODE_RGB565
    `define Resolution_800x480 1 //时钟为 33MHz
```

```
`elsif HW_VGA    //使用 VGA 显示器，默认为 640*480 分辨率，24 位模式
//=====
//可选择其他分辨率和 16 位模式，需用户根据实际需求设置
//代码下方三行和四行设置位模式
//代码下方五行以后连续宏定义部分设置分辨率
//=====
`define MODE_RGB565
// `define MODE_RGB888
// `define Resolution_640x480    1 //时钟为 25.175MHz
//`define Resolution_800x600    1 //时钟为 40MHz
//`define Resolution_1024x600    1 //时钟为 51MHz
//`define Resolution_1024x768    1 //时钟为 65MHz
// `define Resolution_1280x720    1 //时钟为 74.25MHz
`define Resolution_1920x1080 1 //时钟为 148.5MHz
`endif
```

如此，我们便完成了 tft_ctrl_test 模块的修改，模块在工作时会按照配置表中的参数产生 1920*1080 分辨率的 RGB565 格式数据，输出给 SiI9022。

32.2.2 创建顶层封装

修改完模块后，我们还需要创建一个顶层模块。顶层模块的设计也非常简单，只需要对子模块进行例化，同时产生一个高脉冲的 Go 信号即可。需要注意的是，由于所需图像分辨率为 1920*1080，在例化 tft_ctrl_test 模块时，我们需要通过顶层传参的方式，修改 tft_ctrl_test 模块产生的彩条图像分辨率。完整的顶层代码如下：

```
module color_bar_hdmi(
    input clk50M,
    input reset_n,
    output [15:0]TFT_rgb,
    output TFT_hs,
    output TFT_vs,
    output TFT_clk,
    output TFT_de,
    output TFT_pwm,
    output SiI9022_sclk,
    inout SiI9022_sdat,
    output led
);

    wire clk_ctrl;
    wire locked;

    parameter DISP_WIDTH = 1920;
```

```
parameter DISP_HEIGHT = 1080;

tft_ctrl_test
#(
    .DISP_WIDTH(DISP_WIDTH),
    .DISP_HEIGHT(DISP_HEIGHT)
)
tft_ctrl_test(
    .clk_50M(clk50M),
    .reset_n(reset_n),

    .TFT_rgb(TFT_rgb),
    .TFT_hs(TFT_hs),
    .TFT_vs(TFT_vs),
    .TFT_clk(TFT_clk),
    .TFT_de(TFT_de),
    .TFT_pwm(TFT_pwm)
);

reg [20:0]cnt;
reg Go;

always@(posedge clk50M or negedge reset_n)
if(!reset_n)
    cnt <= 0;
else if(cnt <= 499999)
    cnt <= cnt + 1 ;
else
    cnt <= 500001;

always@(posedge clk50M or negedge reset_n)
if(!reset_n)
    Go <= 0;
else if(cnt == 499999)
    Go <= 1'b1;
else
    Go <= 0;

SiI9022_Init SiI9022_Init(
    .Clk(clk50M),
    .Rst_n(reset_n),

    .Go(Go),
    .device_id(8'h72),
    .Init_Done(led),
```



```
.i2c_sclk(SiI9022_sclk),  
.i2c_sdat(SiI9022_sdat)  
);
```

```
endmodule
```

至此，我们便完成了测试工程的设计。对设计进行综合分析，确认无误后便可以开始进行板级验证了。

32.2.3 板级验证

本实验的板级验证环节，主要验证以下几个目标：

1. 能否正确将生成的 bit 文件下载到 ACZ7015 开发板。
2. bit 下载完成后，开发板上 PL 侧 LED 灯 T14 是否常亮。
3. 下载完成后 HDMI 显示器能否实现 8 种不同颜色的彩条显示。

32.2.3.1 系统所需硬件

本次设计所需硬件如下，相关模块资料可以点击超链接查看：

1. [ACZ7015 开发板](#) x1
2. 电源电缆一根（可选）
3. USB-typeC 口电缆一根
4. HDMI 线缆一根
5. 支持 HDMI 显示的显示器一台

32.2.3.2 添加 I/O 约束

打开管脚约束窗口，其中 Package Pin 和 I/O Std 是我们要约束的内容。这里，参考以下管脚约束表即可完成管脚绑定。

表 32-2 管脚分配表

Pin Name	Signal Name	Pin NO.	Pin Name	Signal Name	Pin NO.
FPGA_GCLK1	clk50M	L5	Display_G2	TFT_rgb[7]	J6
FPGA_KEY0	reset_n	R4	Display_G1	TFT_rgb[6]	J7
AUD_I2C_SCL	SiI9022_sclk	J3	Display_G0	TFT_rgb[5]	M3
AUD_I2C_SDA	SiI9022_sdat	K2	Display_B4	TFT_rgb[4]	M2
FPGA_LED0	led	P7	Display_B3	TFT_rgb[3]	L1
Display_R4	TFT_rgb[15]	M4	Display_B2	TFT_rgb[2]	L2
Display_R3	TFT_rgb[14]	K5	Display_B1	TFT_rgb[1]	J1
Display_R2	TFT_rgb[13]	J5	Display_B0	TFT_rgb[0]	J2

Display_R1	TFT_rgb[12]	K3		Display_PCLK	TFT_clk	M1
Display_R0	TFT_rgb[11]	K4		Display_DE	TFT_de	P6
Display_G5	TFT_rgb[10]	H8		Display_BL	TFT_pwm	P5
Display_G4	TFT_rgb[9]	L4		Display_HSYNC	TFT_hs	N1
Display_G3	TFT_rgb[8]	R8		Display_VSYNC	TFT_vs	P1

分配完管脚后还需要为管脚约束电平，完成后如

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Term
All ports (26)												
TFT_rgb (16)												
TFT_rgb[15]	OUT		M4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[14]	OUT		K5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[13]	OUT		J5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[12]	OUT		K3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[11]	OUT		K4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[10]	OUT		H8	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[9]	OUT		L4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[8]	OUT		R8	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[7]	OUT		J6	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[6]	OUT		J7	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[5]	OUT		M3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[4]	OUT		M2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[3]	OUT		L1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[2]	OUT		L2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[1]	OUT		J1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[0]	OUT		J2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
Scalar ports (10)												
clk50M	IN		L5	✓	34	LVCNMOS33*	3.300				NONE	NONE
led	OUT		P7	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
reset_n	IN		R4	✓	34	LVCNMOS33*	3.300				NONE	NONE
SiI9022_sclk	OUT		J3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
SiI9022_sdat	INOUT		K2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_clk	OUT		M1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_de	OUT		P6	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_hs	OUT		N1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_pwm	OUT		P5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_vs	OUT		P1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50

图 32-10 所示:

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Term
All ports (26)												
TFT_rgb (16)												
TFT_rgb[15]	OUT		M4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[14]	OUT		K5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[13]	OUT		J5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[12]	OUT		K3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[11]	OUT		K4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[10]	OUT		H8	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[9]	OUT		L4	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[8]	OUT		R8	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[7]	OUT		J6	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[6]	OUT		J7	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[5]	OUT		M3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[4]	OUT		M2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[3]	OUT		L1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[2]	OUT		L2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[1]	OUT		J1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_rgb[0]	OUT		J2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
Scalar ports (10)												
clk50M	IN		L5	✓	34	LVCNMOS33*	3.300				NONE	NONE
led	OUT		P7	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
reset_n	IN		R4	✓	34	LVCNMOS33*	3.300				NONE	NONE
SiI9022_sclk	OUT		J3	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
SiI9022_sdat	INOUT		K2	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_clk	OUT		M1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_de	OUT		P6	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_hs	OUT		N1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_pwm	OUT		P5	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50
TFT_vs	OUT		P1	✓	34	LVCNMOS33*	3.300	12		SLOW	NONE	FP_VTT_50

图 32-10 引脚分配表

完成引脚分配和约束后，接下来为设计生成 bit，确认设计无误并生成 bit 流后，接下来便可以连接硬件并准备将 bit 烧录到开发板中进行验证了。

32.2.3.3 硬件连接与验证

本次验证设计硬件连接如图 32-11 所示：

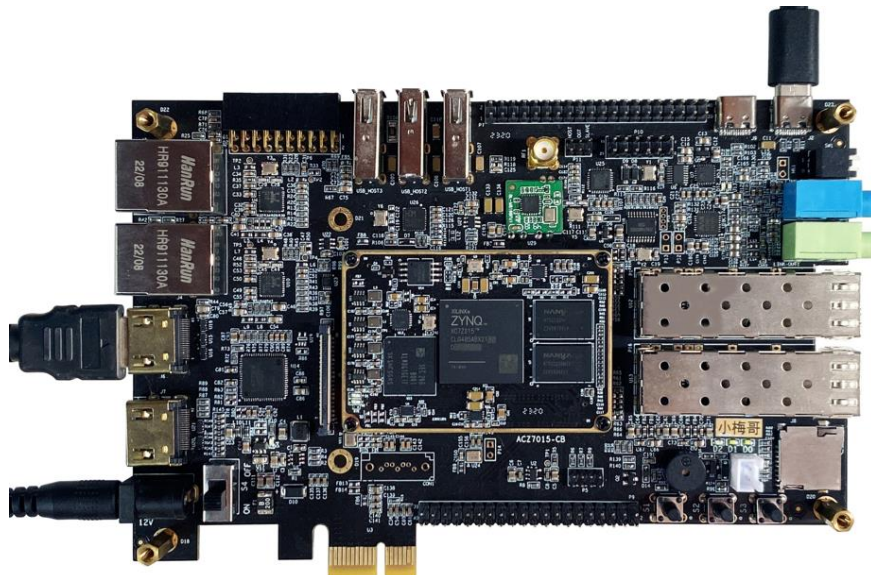


图 32-11 硬件连接图

在连接 HDMI 线缆时，请务必确保线缆一端连接开发板上的 HDMI 接口，一端连接支持 HDMI 显示的显示屏。硬件连接完成后，拨动开发板电源开关到对应供电侧，将 bit 文件烧录到开发板中。等待片刻观察 HDMI 显示屏上的现象，如图 32-12 所示：

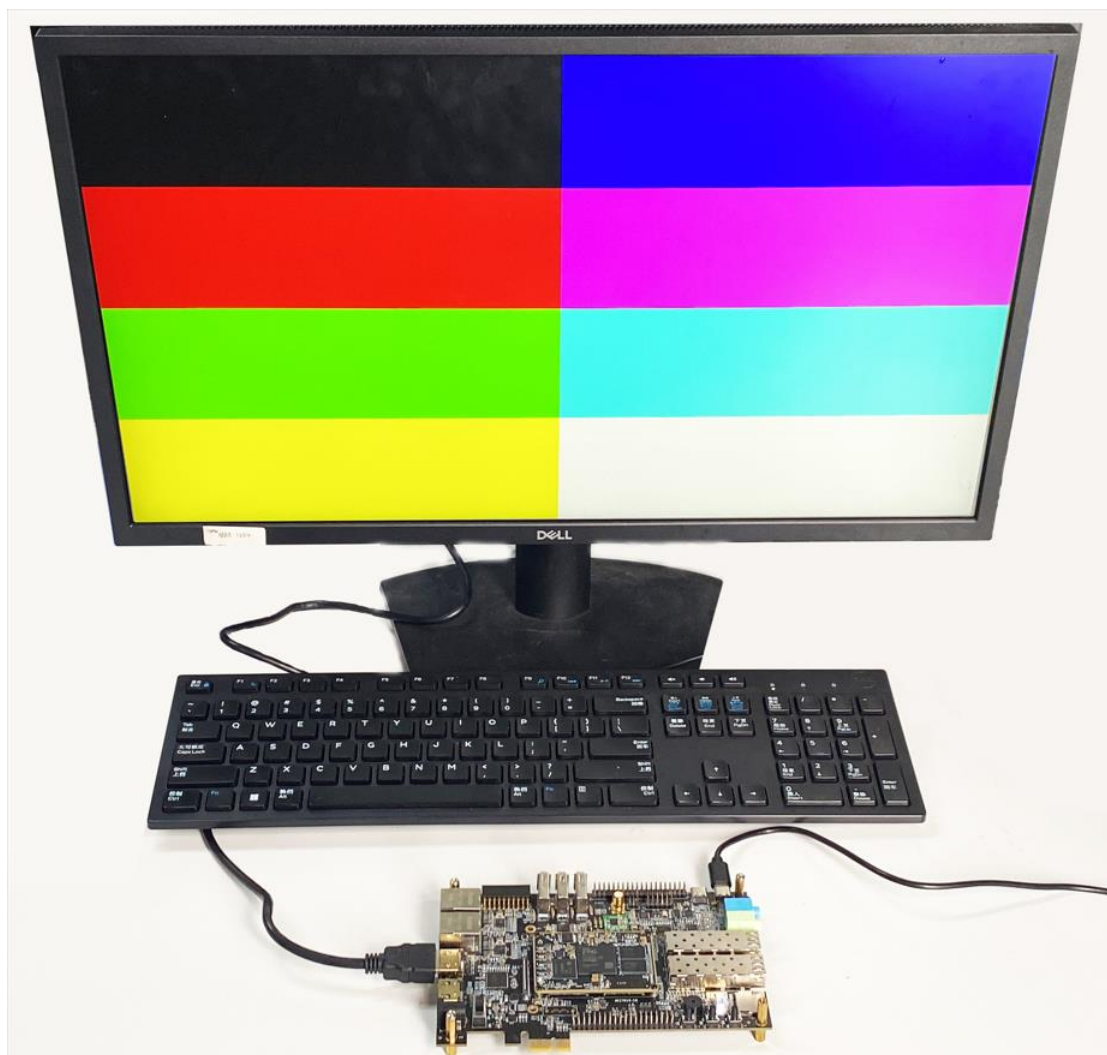


图 32-12 HDMI 显示

可以看到，显示图像分辨率为预期的 1920*1080，显示的彩条与设计所预期的一致，颜色鲜明，层次清晰，说明本次设计成功。

32.3 总结

本章带大家完成了 SiI9022_Init 模块的设计，并通过彩条实验验证了设计模块功能的正确性。基于本次设计，用户在后续的各种图像相关的应用中，只需要按照本节内容最后采取的方法，使用 SiI9022_Init 模块对 HDMI 芯片初始化，同时将用于显示的图像数据和时序信号传输给 SiI9022，就能实现 HDMI 的显示。