基于 FPGA 的自适应屏幕主题的氛围灯设计

队员：李致萱、石宇航、汤玮民

第一部分　设计概述

1.1 设计目的

随着 LED 技术的不断发展，灯光的渲染已被各个电子厂商加入到自己的产品应用当中，而目前 LED 氛围灯的主流实现方法均为软件实现，真正结合显示器的色调变化硬件硬件实现的氛围灯几乎处于空白，由于通过软件视频流的处理方式无法做到与其屏幕连接设备的同步，易造成氛围灯延迟响应，影响用户体验，同时软件实现的硬件电路制造周期长、设计成本高、制成后不易修改，加大了研发成本。因此，为了将 FPGA 的低功耗、精度高、处理速度快、可重复编程配置、模块化设计等优势利用到屏幕氛围灯的研发中，缩短产品的研发周期。本设计基于紫光同创的 PL22G 开发板，使用 PC 上位机与 FPGA 上搭建硬件电路的方法实现了一种屏幕氛围灯同步控制系统。可以最快得出屏幕区域的 RGB 值，并且无延时的传输到 RGB 灯带显示屏幕块区的主色调，同时使用较少的 FPGA 资源，节约研发成本。

1.2 应用领域

本作品可以运用的领域从商业使用的大型 LED 广告牌到家中的电视、投影、显示器等显示设备，均可以通过自适应屏幕主题氛围灯，增强商业广告和家庭显示器的显示效果，提供给用户更好视觉体验。

1.3 主要技术特点

本作品简单便捷且实用性强大。在基于 FPGA 硬件平台上不仅自主设计一种对视频图像边缘区域分区，同步提取视频图像的色度、亮度的算法，而且基于此算法在计算机上使用了一个采集屏幕边缘分区 RBG 数据和与 FPAG 进行串口通信数据传输的上位机程序。在硬件设计方面，我们通过 Altium Designer PCB 设计软件进行了 WS2812 灯带的 PCB 设计，能与紫光同创 PL22G 开发板配合使用，实现视频图像色度、亮度显示。同时通过紫光同创的 PDS 集成开发环境联合 Modelsim 仿真工具，完成了整个系统的仿真验证。

1.4 关键性能指标

（1）本作品中采用的紫光同创的 PL22G 开发板，其核心板主要由紫光 FPGA+DDR3+QSPI FLASH 构成，具有丰富的逻辑资源与接口，适合高速数据通信，视频图像处理，高速数据采集的应用场景。基于 PL22G 开发板在其上面部署视频

图像色度、亮度提取算法，支持视频图像边缘分区数与灯珠数一一对应进行色度、亮度提取。最大分区数可达 300 个分区，同时在算法设计上能够对 UART 串口输入的 RGB 数据进行实时运算，运算速率可支持处理 300 分区以上的数据计算。

（2）在计算机上基于此算法配套了一个上位机软件 AmbiBox.exe，此上位机软件具有 UART 串口发送、屏幕边缘分区、屏幕显示静态、动态画面图像实时捕获 Screen Capture 等功能。支持的视频图像边缘分区数可手动设置，同时还能够选择不同 RGB 色彩编码存储格式进行串口数据传输。

1.5 主要创新点

（1）本作品中优化了一个易于 FPGA 实现的视频图像色度、亮度提取模型，该模型通过电脑上位机软件完成对视频图像进行色度、亮度边缘区域粗分割，为降低后续处理的运算量，将分区数设定为与灯带灯珠数一致，通过分区与灯珠对应，准确将 RGB 值传入灯带，同时通过 FPGA 对区块的 RGB 值进行均值运算得到最佳色度数据。

（2）算法设计上采用紫光同创 PL22G 系列 FPGA 开发板作为视频图像处理的核心，通过研究图像数据传输模块、色彩转换模块、图像存储模块、图像处理模块以及灯带驱动模块，实现硬件化的视频图像色度、亮度提取系统，优化了算法采用移位代替除法的思想，在屏幕分区块时，将每一个区块定义为 32*32 大小，这样在完成累加之后，要得到该区块的平均值便可以直接将累加值左移十位得到目标值，放弃了选择除以（32*32）得到目标值的传统计算方法，且采用 Verilog 全模块化设计，具备良好的可移植性。能实时，准确的视频图像进行边缘分区和主色调提取。系统设计占用资源少，具备低功耗、低时延的特点，符合智能化，小型化、快速高效处理的发展趋势。

（3）采用上位机进行视频边缘 RGB 信息的采集，在边缘区块的分区上可灵活的进行区块大小的设定以及颜色信息的修改。通过上位机+FPGA 的信息处理模式，同时使用 HDMI 输出将视频输出到显示器上，使得本系统还可以用于其他平台基于 FPGA 的设计，且上位机的使用极大地优化了开发的成本。

第二部分 系统组成及功能

2.1 整体介绍

2.1.1 方案介绍

本方案设计主要由取色上位机的设计和 RGB 灯的控制系统两部分组成。取色上位机是基于 QT5 软件开发的控件。 RGB 灯带的控制系统包括 WS2812 驱动芯片和紫光同创的 PL22G FPGA 微处理器。

总设计流程为 PC 端上位机对屏幕进行颜色提取、屏幕边缘色彩分区与主色调分析，FPGA 做为下位机，通过串口通信的方式接收数据，对传入的屏幕色彩数据进行处理，并对 WS2812 RGB 三色灯带进行控制，并且使用 HDMI 视频输出的形式，在显示器上显示原始视频。所以此次设计采用色度 RGB 数据处理与视频显示并行处理，加速系统的处理速度的方案。

2.1.2 系统组成

1、硬件部分组成：紫光同创 PL22G 系列 FPGA 开发板、WS2812 灯带条、带 HDMI 接口的显示器。

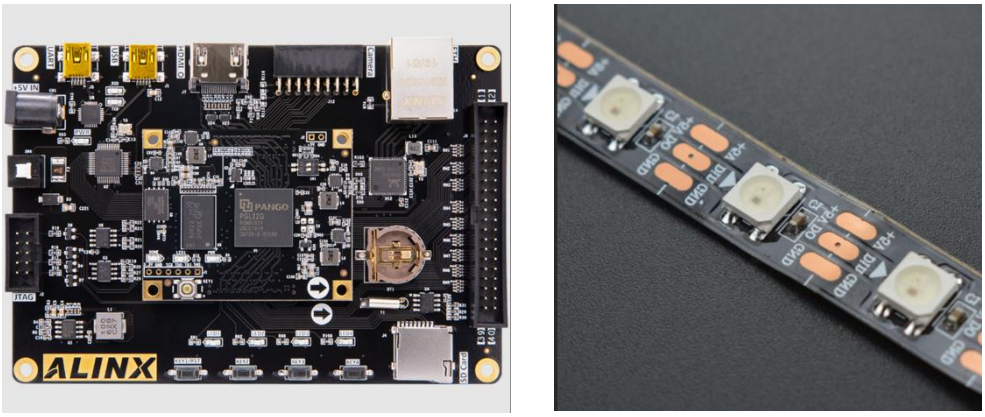


图 1 主要硬件实物图

2、软件部分组成：主要由上位机图像处理模块、FPGA 串口输入模块、串口数据处理模块、色度、亮度提取模块、HDMI 输出模块以及灯带驱动模块组成；

2.1.3 系统整体框图

本设计的 FPGA 设计的整体框图如图 2 所示：

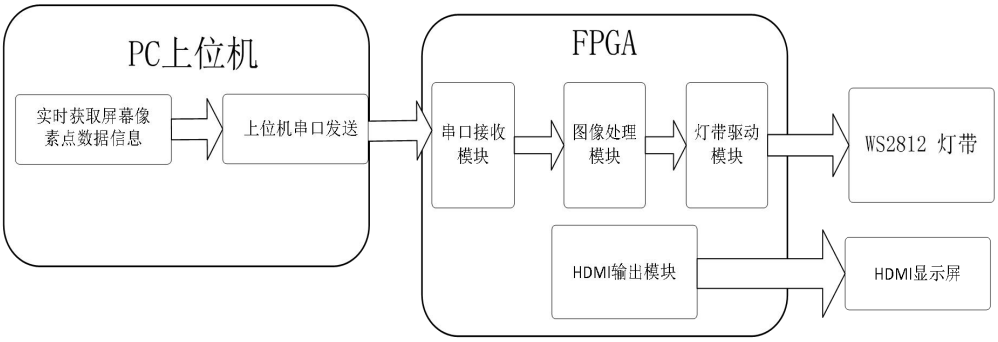


图 1 系统整体框图

2.1.4 各模块介绍

（1）上位机模块介绍

本设计所使用的上位机程序主要用于对屏幕图像进行分区块像素数据采集，后通过串口传输给 FPGA 进行数据处理。上位机主要有以下部分组成：

1、控件功能

屏幕控件 CombiBox

控件 CombiBox 界面主要由灯珠数量与 RGB 编码编辑窗口（Edit）与串口号列表（List）构成。能够给用户提供多种输入配置模式，可根据用户需要选择使用纯列表选择方式，纯文本编辑的方式或者编辑输入与列表选择相结合的方式进行屏幕边缘分区设置。操作页面如图 3 所示：

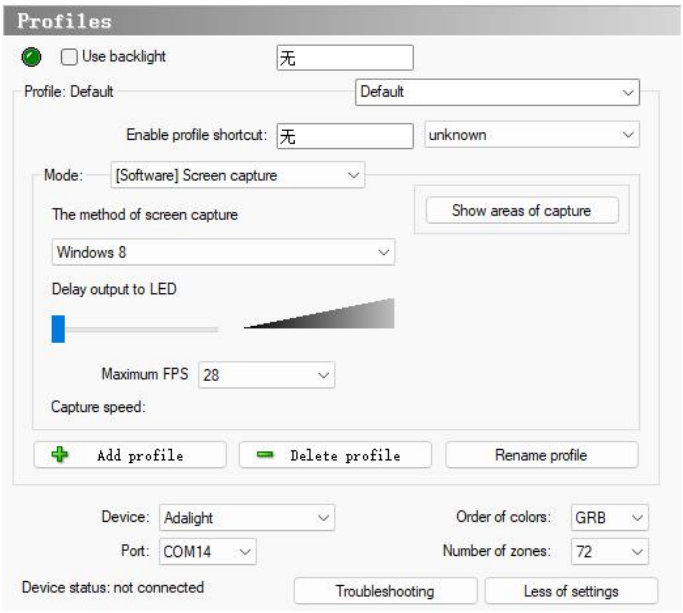


图 3 上位机操作界面

在此设计中用于串口选择、屏幕块区颜色编码选择和灯珠数量选择的下拉窗口设计，通过在图 3 的 Port 选项框中选择 FPGA 板级所对应串口；Order of colors 选项框中设置串口传输的 RGB 颜色编码格式；Number of zones 选项框中设置屏幕周围 WS2812 灯带的灯珠数量。同时我们还设计屏幕边缘块区的分区设置功能与

实时捕获当前屏幕边缘色彩功能。如图 4 所示：

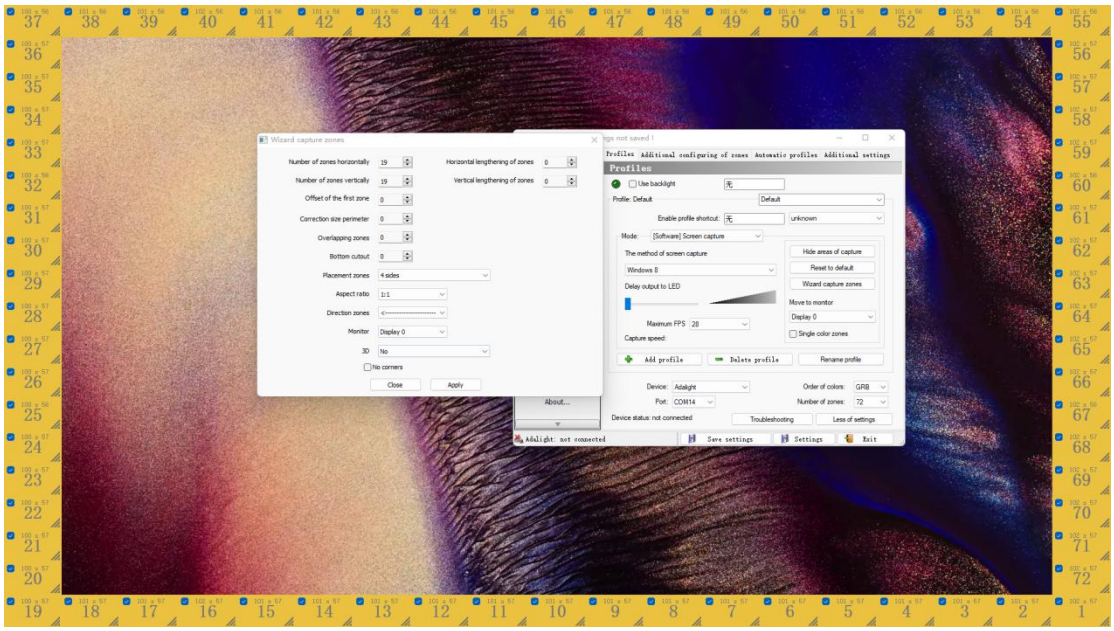


图 4 屏幕分区设置界面

其中屏幕边缘分区主要依据首先通过在图像的边缘区域勾画一周接近屏幕形状的区域网格，如下图 5，划分区块数与设定的灯珠相等。在将划分好的分区进行编号，区域的 RGB 数据与对应的灯珠显示色彩对应。同时该上位机将处理好的分区编号，RGB 数据与分区数据通过串口的方式串行的发送给 FPGA，从而实现对视频图像的分区功能。通过用户所设置的显示器后面的 WS2812 灯带的灯珠数量，来对当前屏幕进行块区分布，此设计更加便于串口输出屏幕块区中的颜色信息，同时经过 FPGA 对串口输入的块 RGB 值进行处理，传输到相应的灯珠上。可以更加快捷、实时的实现颜色信息传输。

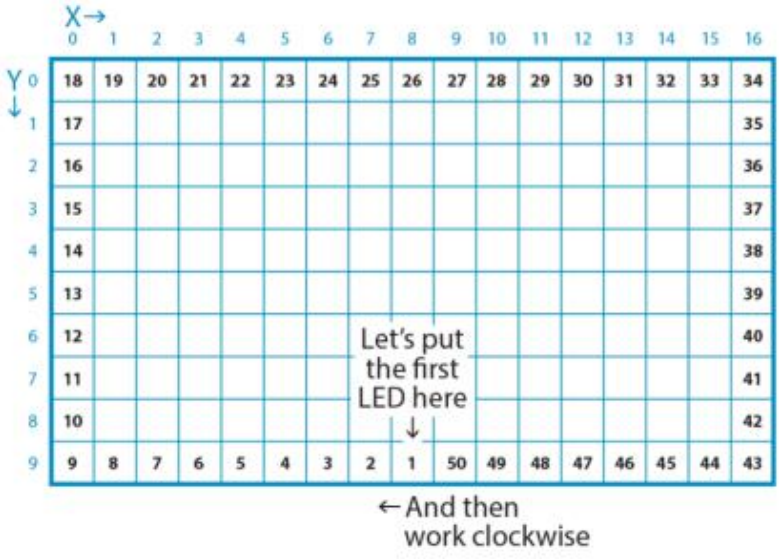


图 5 屏幕网格划分

控件 ImageBox

ImageBox 控件的设计主要用于图像与视频边缘色彩处理。在次设计中用于显示的图像信息经过主色调提取算法后的颜色，通过此控件处理将获取或设置的 RGB 色彩信息打包输出到串口控件 SerialPort 中进行串口发送。

控件 SerialPort

SerialPort 控件用于控制串行端口文件资源，同时实现串口的数据发送功能。通过该控件可以实现电脑上位机与 FPGA 之间的串口通讯，其中默认支持 115200 的串口波特率传输。

（2）串口接收模块块及 RGB 数据整合模块介绍

串口接收模块通过 uart_rxd 端口接收来自电脑上位机采集的图像像素数据，在接收过程中，UART 从数据帧中去掉起始位和结束位，对进来的字节进行奇偶校验，输出 8 位的有效数据，并将输出 8 位数据输入到 RGB 数据整合模块中，每接收一个 8 位的像素数据，将其添加到一个区块的寄存器中，通过三个串接的移位寄存器，产生一段 24 位的 RGB 数据，从而实现屏幕图像不同分区的 24 位 RGB 数据的整合。

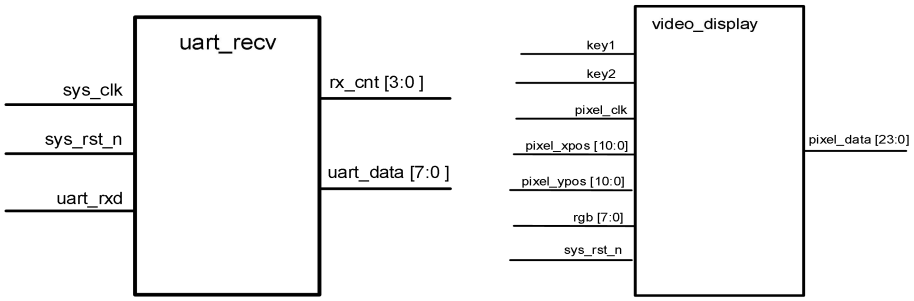


图 6 串口接收模块及 RGB 数据整合模块

（3）图像色度、亮度处理模块介绍

该模块设计通过接收串口数据处理模块处理后的分区像素点信息存储到预设的对应灯珠的区域寄存器中，当该模块采集到到该区域的最后一个像素点，即可得到该区域的总 RGB 值，后通过进行移位计算，得到该区域的 RGB 平均值，当处理完最后一个编号区域的 RGB 数据后，则完成图像数据采集，实现流程如图 7 所示：

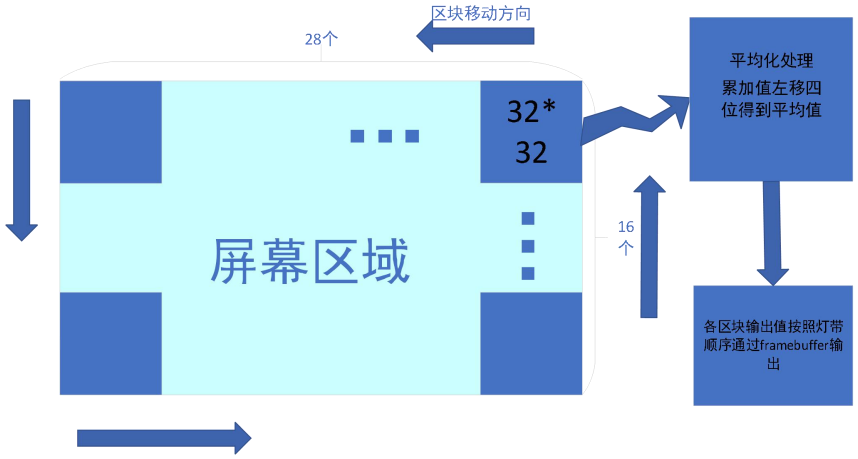


图 7 图像处理模块流程图

图像平均值处理模块 average.v 可以求取每个区块的 RGB 平均值，通过模块水平方向数据输入端口 pixel_xpos 和垂直方向数据输入端口 pixel_y 输入图像数据，每一区块包含 N 个区域；对于垂直方向上的区块，预设 N- 1 个移位寄存器模块，N- 1 个移位寄存器模块首尾相连，每个寄存器模块包含 32 个寄存器；对于水平方向上的区块，预设 1 个移位寄存器，该移位寄存器包含 N- 1 个寄存器；在区域 RGB 平均值计算过程中，利用 3 个移位寄存器模块计算垂直方向上的区块 RGB 平均值，利用移位寄存器模块计算水平方向上的区块 RGB 平均值；具体地，对于水平方向上的区块，将连续 N 个区域 RGB 平均值存储到移位寄存器中，在该区块的最后一个区域的最后一个像素即得到对应区块的 RGB 平均值；对于垂直方向上的区块，将第一次区域值送入移位寄存器模块中，经过 32 次移位后，同时处理当前区域值与同一列的上一个区域值 ，当处理到该区块的最后一个像素时，即可得到区块的 RGB 平均值。

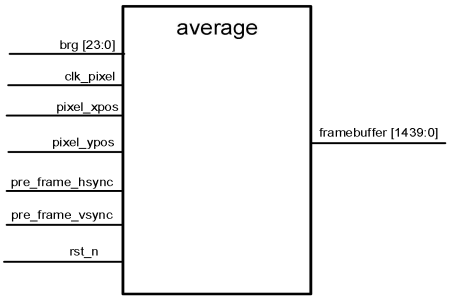


图 8 平均值计算模块 average.v

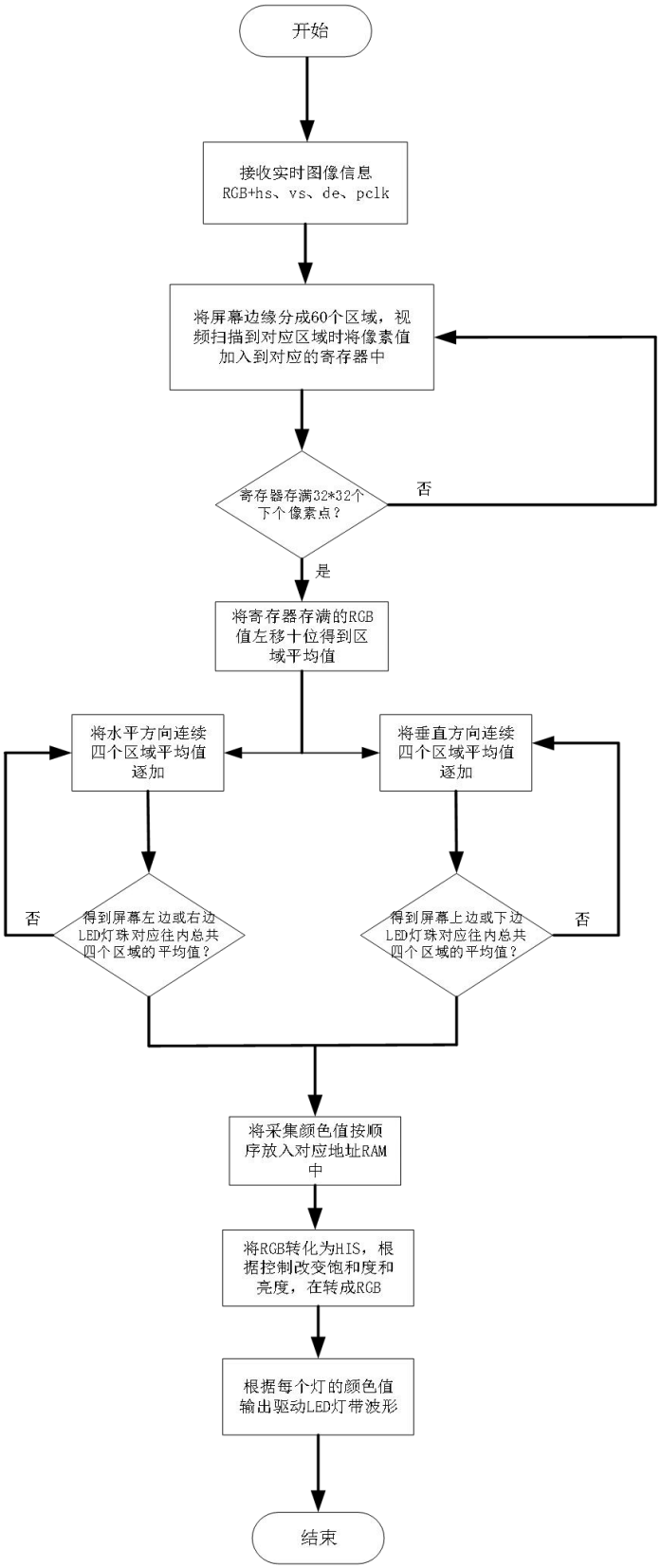


图 9 色度、亮度提取模块设计流程图

（4）HDMI 输出模块介绍

HDMI 视频数据输出模块采用 TMDS（Transition Minimized Differential signal），最小化传输差分信号的数据传输原理。HDMI 中的 TMDS 传输系统分为

两个部分：发送端和接收端。TMDS 发送端收到 HDMI 接口传来的表示 RGB 信号的 24 位并行数据（TMDS 对每个像素的 RGB 三原色分别按 8bit 编码，即 R 信号有 8 位，G 信号有 8 位，B 信号有 8 位），然后对这些数据和时钟信号进行编码和并/串转换，再将表示 3 个 RGB 信号的数据和时钟信号分别分配到独立的传输通道发送出去。接收端接收来自发送端的串行信号，对其进行解码和串/并转换，然后发送到显示器的控制端。与此同时也接收时钟信号，以实现同步。从而实现数据的输出。

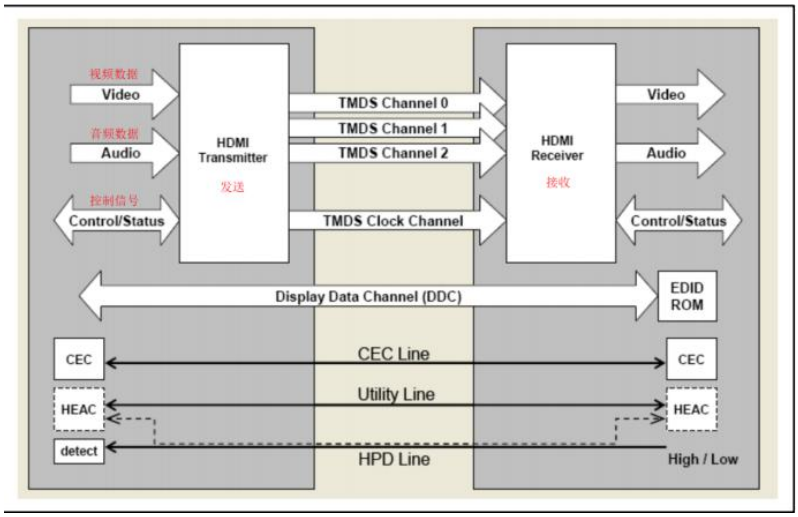


图 10　HDMI 数据传输框图

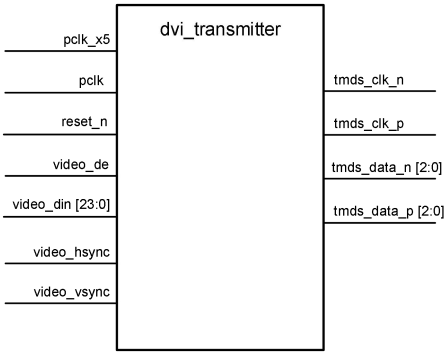
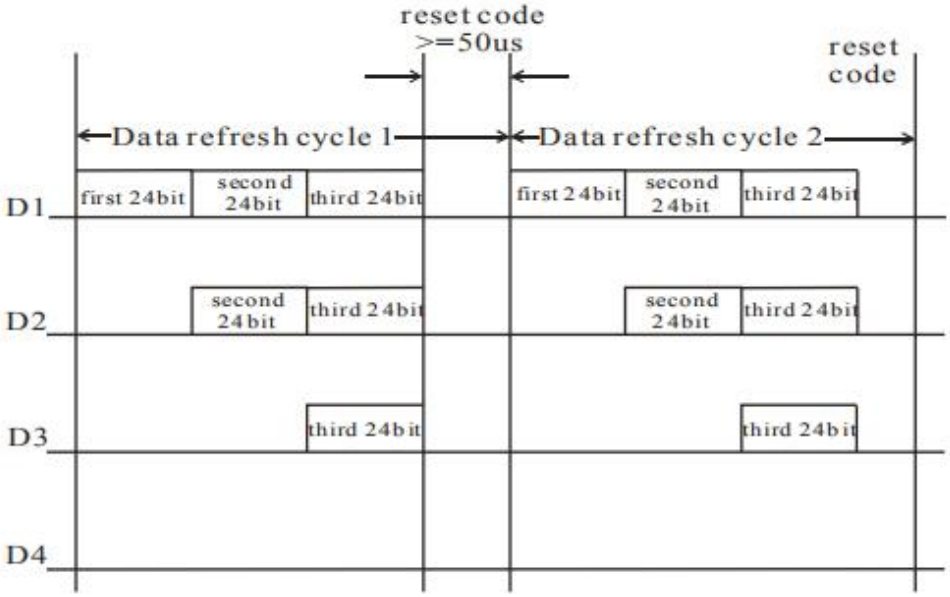


图 11　HDMI 输出模块

（5）WS2812 灯带驱动模块介绍

本此设计使用的 WS2812 灯带采用的是一个及控制电路和发光电路于一体的智能外控 LED 灯，其每个元件即为一个像素点，像素点内部包含了数字接口数据锁存信号整形放大驱动电路、高精度内部振荡器及可编程电流控制部分。通过单线归零码的数据协议进行通讯，由芯片的 DIN 端接收从控制器传输过来的像素数据，将传入的 24 位数据被第一个像素点提取后，送到像素点内部的数据锁

存器，剩余的数据经过内部整形处理电路整形放大后通过 DO 端口开始转发输出给下一个级联的像素点，每经过一个像素点的传输，信号减少 24bit。其驱动时序图如下图 12：



Note: The data of D1 is send by MCU,and D2, D3, D4 through pixel internal reshaping amplification to transmit.

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

图 12　WS2812 驱动时序

在程序设计上 WS2812 灯带驱动模块由 RGB_Control.v 和 RZ_code.v 两个模块组成。将图像色度处理模块移位运算完成后的 RGB 数据值传输到 RGB_Control.v 模块中，将总的 RGB 数据值转换成每段 24 位的 RGB 数据（即一个 D）输出给 RZ_code.v 模块，在 RZ_code.v 模块中按照图 13 所示 D1（第一颗 LED）收到 3 段数据（3×24bit），截取第一段 24bit 数据，剩下两段数据（2×24bit），交给 D2，D2 同样截取一段，剩下最后一段，交给 D3 按照这样的读取方式，每 24bit 数据从高位到低位进行单极性归零码判断是高电平还是低电平后输出，最终由 RZ_data 端口输出驱动波形。

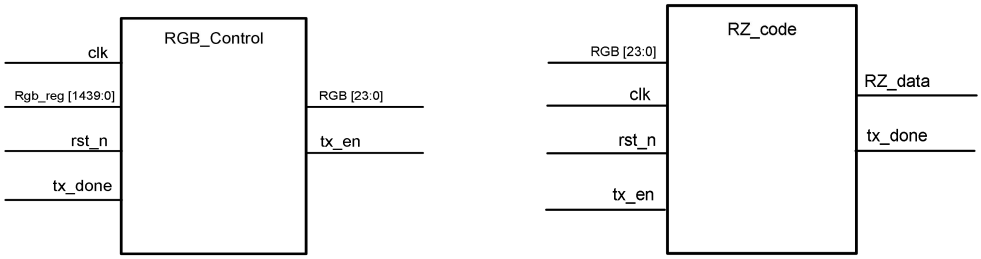


图 13　WS2812 主要驱动模块

第三部分　完成情况及性能参数

如前所述，在完成了系统整体框架分析以及模块介绍后，本部分着重讲述重要模块的仿真及其时序分析和实物效果的完成情况，具体介绍如下所示：

3.1 重要模块 Modelsim 功能仿真及时序分析

Modelsim 是公司推出的一款基于硬件描述语言的仿真软件，它为用户提供友好的调试环境，同时它编译仿真速度快，编译代码与平台无关，能够有效地保护核，它是当前进行设计的首选仿真工具软件。它全面支持和语言的标准和功能调用与调试。利用仿真各模块的波形易于观测，仿真效率高。本设计采用紫光同创 PDS 集成开发环境与 Modelsim 仿真工具对主要模块进行功能仿真验证。

3.1.1 串口接收模块仿真

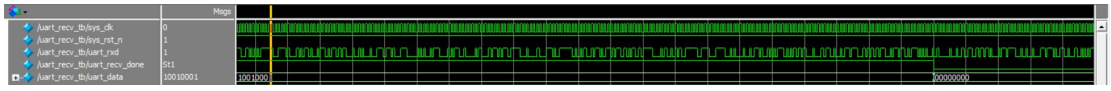


图 14 uart_recv.v 模块仿真图

仿真分析：

uart_recv 模块：通过 uart_rxd 随机赋值模拟串口传入的数据，当 sys_rst_n 等于 1 时，根据接收数据计数器来寄存 uart 接收端口数据，数据接收完毕后给出标志信号并寄存输出接收到的数据，接收数据计数器计数到停止位时将接收完成标志位拉高。

3.1.2 色度均值化处理模块仿真

图 15　average.v 模块仿真图

仿真分析：

由于实际的区块为 32*32，这对于仿真来说是不合适的，因此，在进行仿真前，先将代码中的区块大小减小，本次仿真改为了 4*4 大小，之后再进行仿真，本此只选取了四个区块值计算后的输出，如图验证，经计算对输入取平均值之后，将场同步信号拉高，计算输出信号的值，发现如理论值相同，仿真完成。

3.1.3 HDMI 输出模块仿真

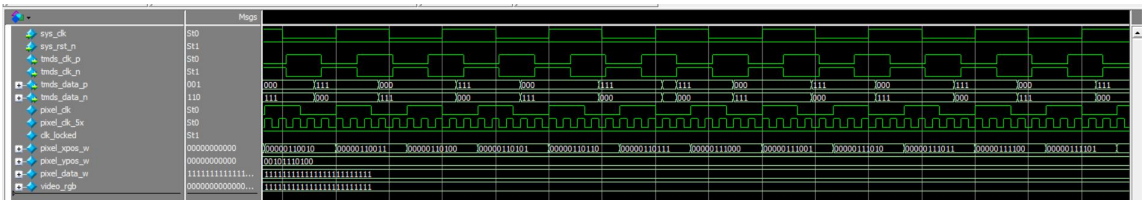


图 16 dvi_transmitter_top.v 模块仿真图

仿真分析：

HDMI 输出模块仿真波形如上图 16 所示，RGB 信号遵循 TMDS 最小化差分信号传输时序输出，HDMI 模块将 24 位的 RGB 图像信号通过对 R、G、B 三个颜色通道进行编码，将 8bit 编码成 10bit 数据，在进行并串转换得到输出串行数据 tmds_data_p 和 tmds_data_n 输出，由仿真可知输出数据 tmds_data 时序正确。

3.1.4 WS2812 灯带驱动模块仿真

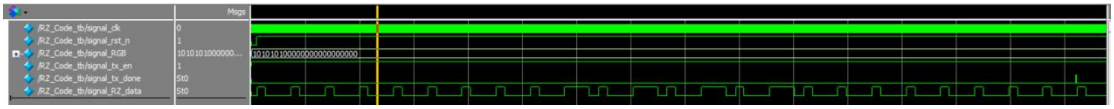


图 17　RZ_Code.v 模块仿真图

仿真分析：

RZ_Code 模块：给 signal_RGB 随机赋值，且当 signal_rst_n=1 时，进行数据解析并输出输出数据为单极性归零码(Return Zero Code)，当处理完 24 位 signal_RGB 后 signal_tx_done 拉高。然后对下次已处理输入的 signal_RGB 进行解析如此循环。

本设计使用 Verilog 硬件描述语言实现各个模块的设计，通过 Modelsim 仿真，观察各个模块的仿真波形并不断修改进行验证。当各个模块验证完成后总和成一个完整系统后进行总体仿真，纠除模块之间的互联错误、系统总线不协调、逻辑不匹配等问题。确认仿真无误后，最终利用 FPGA 硬件实现。

3.2 实物效果完成情况及性能

此次设计的 FPGA 的硬件验证，通过将视频图像的 RGB 图像信息通过串口数据传输的形式输入到 PL22G 开发板中，通过 FPGA 处理后，产生 WS2812 灯带驱动波形，点亮灯带，同步变化，实现流光溢彩效果。具体完成情况如下所示

3.2.1 系统硬件整体情况

本次设计的基于 FPAG 的自适应屏幕主题氛围灯设计，系统整体如下图 18 所示：

图 18 自适应氛围灯 FPGA 硬件实现平台

3.2.2 视频效果完成情况

本次设计的自适应氛围灯 FPGA 硬件实现平台的最终视频效果呈现如下图 19 所示,视频的动态演示效果可观看实物视频。

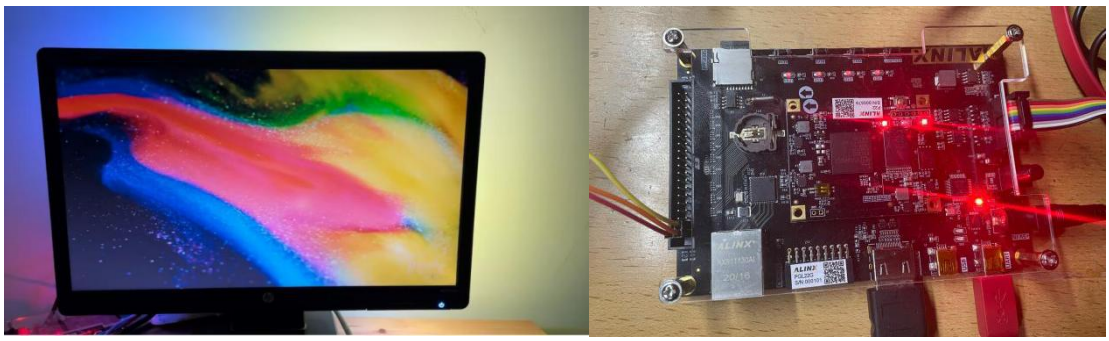


图 19 自适应氛围灯视频效果图

第四部分 总结

4.1 可扩展之处

作品采用的 PL22G-6MBG324 作为主控芯片，使整个视频图像色度、亮度提取显示系统具有小型化、集成化、智能化、实时性好、灵活性高等特点，但此次设计工作在许多环节上还有待于进一步完善和改进。有以下改进点：

1、在输入像素信号之前加入高斯噪声滤波器。滤除高斯噪声，有效地消除明显的大噪声。可通过滤波模块对即将输入的像素进行预处理，降低干扰信号的对后续处理的影响。

2、加入 HDMI 输入模块。由于本次设计所采用的开发板不具备 HDMI 数据输入的功能，所以本次设计使用了电脑上位机，进行像素数据的采集，但是效果较为一般，后续可通过加入 HDMI 的视频输入模块，增强系统的稳定性，使视频效果更为流畅。

3、对主色调提取算法进行改进。在色度提取算法上还可以对像素信号的频

率进行判断，在像素信号变化频率高的部分不做处理直接输出，在像素信号变化频率地的部分进行色度、亮度提取。减少像素的处理量，提高处理速度，同时处理后的图像不易紊乱，使流光溢彩的效果更真实，精彩。

4、受限于屏幕大小，本次设计我们的区块数在边缘分区上并没有超过 256 个，因此，在此基础上，我们将会采取继续增加灯珠的方法，可以通过采用更大的屏幕、选择更密集的灯带，从而达到分区数大大增加的目的。

4.2 心得体会

通过本次 FPGA 创新创业设计大赛，我们不仅收获了丰富的项目经验，也从中学到了很多专业知识，拓展了我们的知识面，以下是我们本次项目的心得体会：

1、心理素养是影响竞赛的关键因素。竞赛不仅比技术，也比心理素养。技术再高的人假如欠缺过硬的心理素养，将会直接影响到自身的发挥。本次比赛，我们选择了紫光同创赛道的电视氛围灯设计选题。在初步确定题目后，我们开始讨论项目方案以及其中的算法设计。虽然之前没有做过此类的选题，但俗话说万变不离其宗，通过在网上查阅资料与老师的指导和队员们的共同努力下，稳定心态最终有所突破。例如，本次实验，在得知开发板中没有 HDMI 的输入模块后，我们很快讨论出选用摄像头或者可以采集像素点数据的相关软件来进行数据采集的方式，巧妙地解决了本次项目的一个问题。在寻找主色调提取算法，由于没听说过相关名词，还以为相关算法会很难实现，后来经过学习后发现平均化处理便可达到很好的效果。所以说，心理素质对我们每一个人来说都很重要，阵脚不乱才能稳步前进。

2、团队精神至关重要。在竞赛中，团队合作尤其重要，大家都会遇到不同的问题，可以相互讨论，分工协作，共同促进，只有拥有这样的团结精神，才能在竞赛中“厚积薄发”，取得更好的成绩。

3、讲速度、更要讲细心。竞赛比的就是完成的效率快不快，以及程序设计过程中细不细致和系统的稳定性高不高。竞赛中失误大部分都是不细心所导致的。选好题目时要认真读懂题意，弄清题目所要实现的功能，这就要求在编程时不仅速度要快，而且不能遗漏触点和指令，一步一步的编，但不能盲目地追求速度，这样才能保证编程既有速度又有质量。本次实验我们就犯了这样的错误，在清楚平均化算法之后，直接就上手去编写，结果不出所料，在编译时面对一堆的红色 error 傻了眼，只能细心的一个个改正，浪费了较多的时间。

4、要学会自学。这是我们作为一名大学生，必须要具备的能力，自学的能力是我们要具备的基本素养，遇到难题和不懂的问题肯定要自己去查资料，不要由于没有学过而灰心而失去信念，每个同学在课堂上学的东西总是有限的，许多学问是要靠自己去学习和积累。这次竞赛我收获最大的就是提高了我们的自学能

力以及查阅文献的能力。

通过参加这次竞赛，不但使我们在自己的专业技能上有所提升，并且明白竞赛不是一个人的战斗，是靠团队的整体力量取得成功。在竞赛过程中不断的培养团队协作的能力，注重性格的磨合，在比赛中培养敢打敢拼，不怕困难，不怕辛苦的精神。此外还要对本次比赛指导我们的老师表示真诚的感谢，老师给我们提供了极大的帮助，在我们遇到困难时，总是认真解答我们的疑问，并给出正确的见解，帮我们一次次解决问题。同时，我们还要感谢紫光同创给予我们一次机会参加这次竞赛，给我们提供了一个锻炼自我的平台，让我们在知识竞赛中不断提升自我，增强自身竞争力，我们也衷心祝愿国产芯片公司紫光同创越办越好，为中国芯片事业发挥巨大作用！以上便是本次竞赛的心得体会。

第五部分　参考文献

[1] 谭会生，张昌凡. EDA 技术及应用. 4 版. 西安：西安电子科技大学出版社,2011.

[2] 夏宇闻. 复杂数字电路与系统的设计技术[M]. 北京: 北京航空航天大学出版社，2003.

[3] 杨映辉. 基于 FPGA 的 SDRAM 控制器的设计[J]. 电子器件，2006. (2):582-584.

[4] 林明. 基于 FPGA 的数字图像显示系统[J]. 单片机与嵌入式应用，2002. (9):21-23.

[5] 杨海诚，洪景新，陈辉煌. 一种基于颜色跟踪的彩色边缘检测算法[J]. 厦门大学学报(自然科学版)，2006. 45(1):60-62.

[6] 聂絮飞，唐志强 . 探析“六基色”彩色显示的真伪. 现代显示，2006. (10): 83-86.

[7]杨璟，朱雷. 基于 RGB 颜色空间的彩色图像分割方法. 计算机与现代化，2010，(8) : 147-149+171.

[8] 刘战杰，马儒宁，邹国平，钟宝江，丁军娣. 一种新的基于区域生长的彩色图像分割算法.山东大学学报(理学版)，2010 (07) : 76-80.

[9] Hongbin Sun.Nanning Zheng.AnEfficient Motion Adaptive De-interlacing and its

VLSI Architecture Design.IEEE Computer Society Annual Symposium on VLSI，2008,455-458.

[10]王强．一种实时图像处理硬件平台的设计与实现:[硕士学位论文].北京:北京交通大学，2009.

第六部分　附录

1、重要代码：

图像数据处理模块

```
module average   //average.v
(
    //module clock
    input               clk_pixel       ,   // 模块驱动时钟
    input               rst_n           ,   // 复位信号

    //图像处理前的数据接口
    input               pre_frame_vsync ,   // vsync 信号
    input               pre_frame_hsync ,   // hsync 信号
    input               [23:0]brg,
    input               pixel_xpos,
    input               pixel_ypos,
    //图像处理后的数据接口
    output  reg             post_frame_vsync,   // vsync 信号
    output  reg             post_frame_hsync,   // hsync 信号
    output   [1439:0]framebuffer       // 输出 60 个区块像素数据
);

    parameter           nblocks=60;     //区块个数
    parameter  [0:11*nblocks-1]  startx={11'd0,11'd0,11'd0,11'd0,11'd
0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,
11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11
'd0,11'd9,11'd59,11'd109,11'd159,11'd209,11'd259,11'd309,11'd359,11'
d409,11'd459,11'd509,11'd559,11'd609,11'd659,11'd709,11'd759,11'd759,
11'd709,11'd659,11'd609,11'd559,11'd509,11'd459,11'd409,11'd359,11'd
309,11'd259,11'd209,11'd159,11'd109,11'd59,11'd9 };

   parameter  [0:11*nblocks-1]  starty={11'd1274,11'd1229,11'd1184,1
1'd1139,11'd1094,11'd1049,11'd1004,11'd959,11'd914,11'd869,11'd824,1
1'd779,11'd734,11'd689,11'd644,11'd599,11'd554,11'd509,11'd464,11'd4
```

```
19,11'd374,11'd329,11'd284,11'd239,11'd194,11'd149,11'd104,11'd59,11
'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd0,11'd
0,11'd0,11'd0,11'd0,11'd0,11'd1242,11'd1242,11'd1242,11'd1242,11'd12
42,11'd1242,11'd1242,11'd1242,11'd1242,11'd1242,11'd1242,11'd1242,11
'd1242,11'd1242,11'd1242,11'd1242};
    reg                 [23:0]a_brg;
    reg                 pre_frame_hsync_d;
    reg                 pre_frame_vsync_d;
    reg         [33:0]   accumulator  [0:nblocks];
    reg         [1439:0] a_framebuffer ;
   Integer               i;

always@(posedge clk_pixel or negedge rst_n) begin

    for (i=0; i<= nblocks-1; i=i+1) begin
         if(pixel_xpos >= startx[11*i+:10] & pixel_xpos <
startx[11*i+:10]+6'd32 & pixel_ypos >= starty[11*i+:10] & pixel_ypos <=
starty[11*i+:10]+6'd32)
        // We are a part of block bn. Accumulate the color info.

          accumulator[i] <= accumulator[i]+ a_brg;
  end
       a_brg <= brg;

end

always@(posedge clk_pixel ) begin
integer j;
   if (pre_frame_vsync == 1'b1 )
           for (j=0; j <= nblocks-1; j=j+1) begin

             a_framebuffer [j*24+:23] <= accumulator[j][33:10];
             accumulator[j] <=0;
           end
end

//延时 2 拍以同步数据信号
always@(posedge clk_pixel or negedge rst_n) begin
    if(!rst_n) begin
        pre_frame_vsync_d <= 1'd0;
        pre_frame_hsync_d <= 1'd0;
    end
```

```
    else begin
        pre_frame_vsync_d <= pre_frame_vsync  ;
        post_frame_vsync  <= pre_frame_vsync_d;
        pre_frame_hsync_d <= pre_frame_hsync  ;
        post_frame_hsync  <= pre_frame_hsync_d;
    end
end

assign framebuffer = a_framebuffer;

endmodule
```

2、WS2812 灯带硬件原理图及 PCB

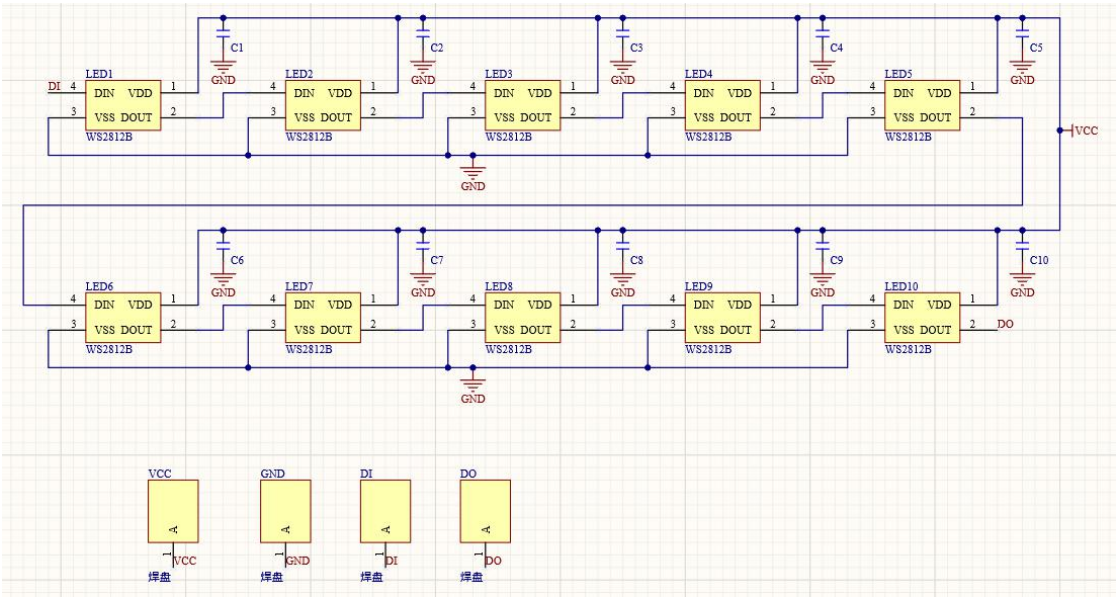


图 20 WS2812b 灯带原理图

2.2 WS2812 灯带 PCB

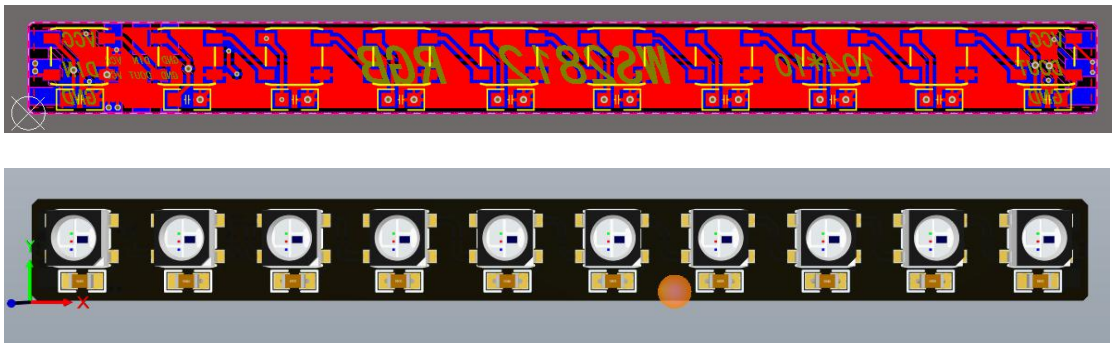


图 21 WS2812b 灯带 PCB 图