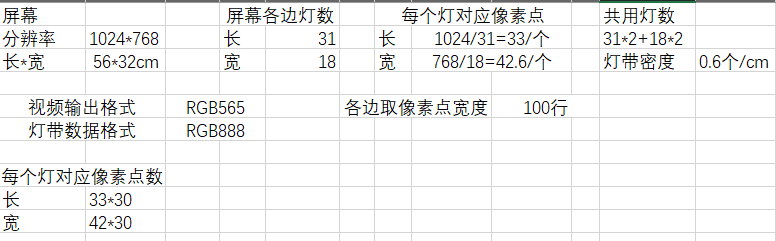
项目综述

本项目基于紫光FPGA实现电视氛围灯效果。即通过FPGA对输入的图像进行处理，获得屏幕四周的色彩信息，在屏幕上输出获得的图像，并将该色彩信息发送到WS2812灯带上来实现电视氛围灯效果。

项目的参数：



视频像素输出是逐行扫描获得的，一个灯30行的数据是需要提前做好处理的。也需要考虑RAM的大小来设计数据处理方式。RAM中只能写一次不能写了再加。

同时上下和左右边的处理原则不完全相同。

只能完成一帧的输出后，才能将灯带信号输出----原因是最下侧屏幕的数据只能完成一帧输出后才能得到。

一帧完成后，对数据处理时，得到的数据是存在FIFO中，在由FIFO发送给灯带。数据的顺序是：下侧、右侧、上侧、左侧。

灯带的绕向是逆时针，灯带头（进数据的地方）在屏幕左下角，屏幕左侧。灯带尾同样也在左下角，不过是在屏幕的下侧。

* 1. 主要技术特点

1、硬件的方式对视频进行处理，处理时延低，整个灯带的最小更新时间小于20ms，本项目采取每八个视频帧更新一次，时延为130ms。

* 1. 关键性能指标

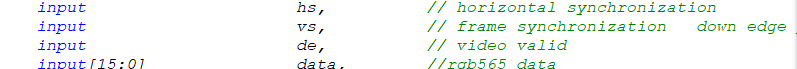
灯带更新时延，本项目采用60fps的视频输入，共98个灯，灯带更新时延为19.7ms，其中16.6ms为显示一帧视频用时，2.94ms为灯带通信时间。

视频分区为31\*18，共558个视频分区。视频分辨率为1024\*768，显示器尺寸为56cm\*32cm

具体的设计方式：

设计为8个视频帧更新一次灯带信息。所以只有需要输出的视频帧才会采集信息，进行计算，否则所以模块全在复位状态。

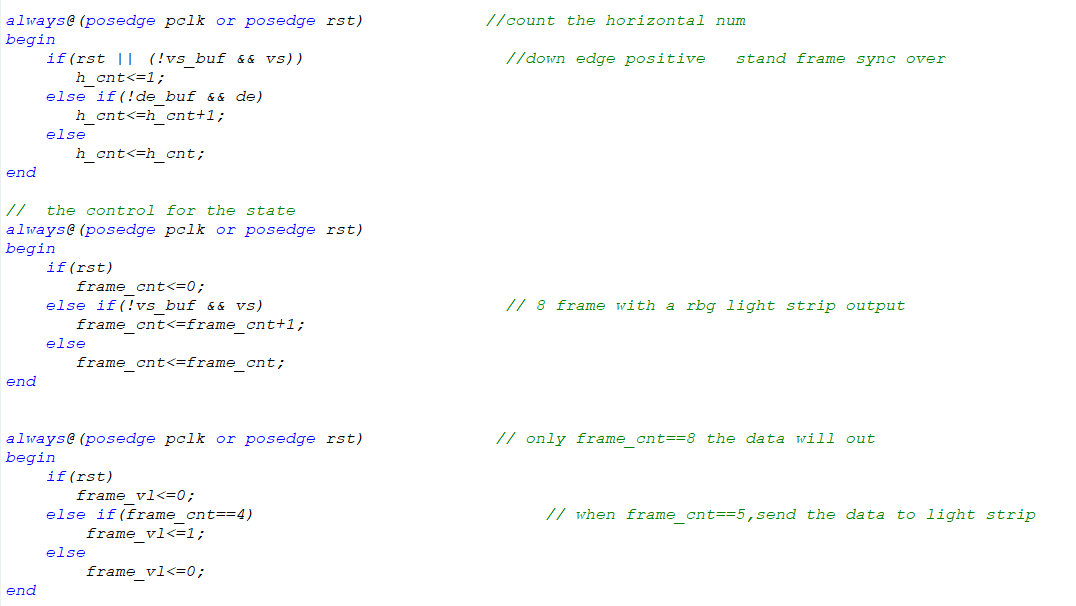
上、下、左、右各有一个数据RAM，分不同的读写，存储数据格式也不一样。



Hs:行同步信号

Vs：帧同步信号，

De:视频有效信号-----为高时视频有效（视频格式其中有无效的仅用于通信的数据）

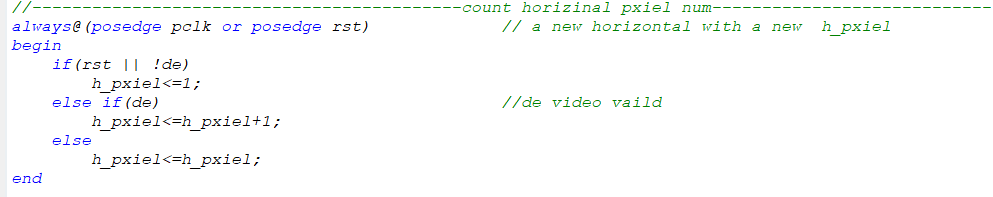


H\_cnt:行计数（上次无效视频，本次有效，代表进入新行）

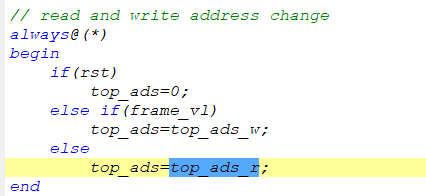
Frame\_cnt:帧计数（上次无效本次有效，帧计数加1）

Frame\_v1:要发数据那一帧

H\_pxiel:像素点，当视频有效时，每个时钟都是一个像素点。



通过Frame\_v1来控制RAM的读写（只有在发送帧是写，其他帧都在读）。



**数据处理部分**

注意RAM存数据地址要自己控制，

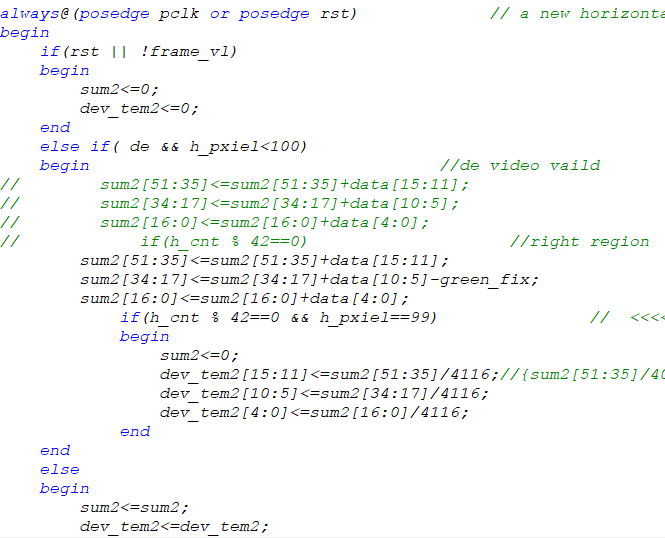
对于屏幕上部区域，定义一个32位的sum1 RGB分别为11 11 10位。把一行33个像素点的灯数据和存到ram中。1行有31个灯，有100行。

底部区域基本相同，和顶部就是顺序不一样。

因为写代码时候脑子一抽，左右弄反了，不用看左右，看实践的区域right\_region<100 left\_region>924。

对于right\_region处理方法：

他一次就可以的搞到一个灯的所有数据，所以都是先把所以的像素点加起来，然后平均了在存到RAM中的。绿色进行了一个参数修正，因为摄像头拍出来色不对，容易把黑色存成绿色，所以加了一个参数修正（所以区域都加了）。



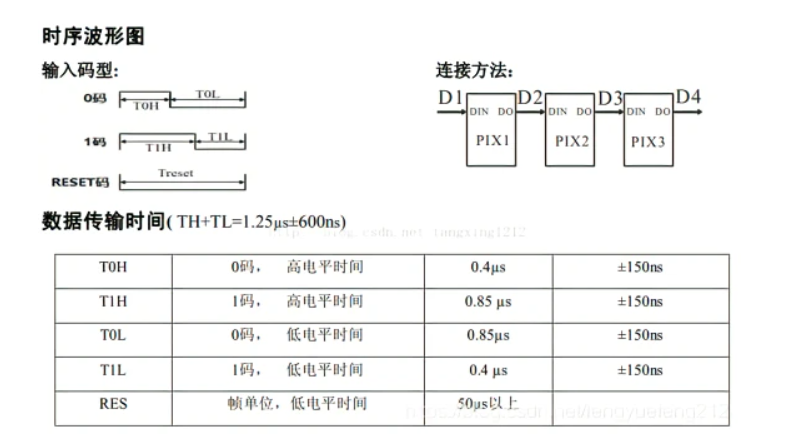
Left\_region过程与此相同。

**数据输出过程：**

是先处理底部然后分别是右侧、上侧、左侧。按照顺序存FIFO中，等待发送。只要不是信号处理帧且FIFO中有数据都可以发。

发送数据的时候，最开始采取通过单片机来接受数据然后，控制串口，但是通信的时候会丢数据，丢一个灯带就全乱了。一直因为这个卡了很久，最后改成FPAG直接给灯带发。灯带的通信方式是归零码通信，单独写了一个电平转换的模块。

数据协议采用单线归零码的通讯方式，像素点在上电复位以后，DIN端接受从控制器传输过来的数据，首先送过来的24bit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。像素点采用自动整形转发技术，使得该像素点的级联个数不受信号传送的限制，仅仅受限信号传输速度要求



一次发一个灯，1个灯发完了，发下一个，每个灯的发送从高位到低位。

发完一个灯，再从FIFO中读下一个，这中间是有通信的。

长时间给一个reset，代表本次更新完成。

通信转换模块里，发完24个bit的数据就给出一个发送完成标志，根据这个标志去在读一次FIFO就可以再发一次数据。

WS2812通信可以参考这篇文章

原文链接：<https://blog.csdn.net/lengyuefeng212/article/details/109706246>