浙江大学

实验报告

智能自行车助手

院系：信息与电子工程学院

课程：数字系统

组别：第七组

摘要

在如今的大学生活中，自行车是最为常见的交通工具，但由于校内人员众多，所以在教学楼附近往往会停放着大量的自行车，而若不加以管理，则整体的摆放会看起来十分混乱，为了改善停放情况，保安叔叔会帮助同学们重新摆好自行车，并且对于停放位置不正确的自行车进行移位，进一步提高空间利用率和改善整体的停车场外观。

但由于车实在是太多了，而且可能来的时候车还很少，但是下课之后发现整个都停满了车，所以在寻找上会存在困难，况且，自行车也有可能会被移动位置，那就更不好找了。这在一定程度上浪费了时间，而且，校园开放后人员混杂，也出现过同学的自行车被校外人员随意骑走的情况，那么寻找起来就更困难了。

基于此，我们计划制作一个智能自行车助手，来解决上述问题。

1.1 设计目的：

整个项目的设计目的就是解决上述问题，当然，最终的目的，是通过本项目来增进我们对于数字系统，FPGA，通信协议，硬件特性等方面的理解与见识，在不断解决问题中提升自我。

1.2 设计要求：

整个项目分为两个端：自行车端与移动端，各使用一块 Ego1 开发板。

自行车端：

自行车端主要的功能就是自行车码表，以及将定位数据发送到移动端。

自行车码表：能够清晰地显示速度，航向，经纬度信息，界面清晰简洁，便于在骑行过程中操作和查看信息。

移动端：

移动端需要显示出自行车和移动端本身的相对位置，即能够通过移动端找到自行车，以及显示两端的定位信息，作为辅助判断，并且在自行车被移动的时候能够发出提醒。

1.3 项目分工：

xxx：整体架构设计，LCD 通用驱动，显示屏问题解决，字符显示，地图缩放渲染模块设计，OLED 滚动显示设计，显示界面设计，硬件选型及采购及组装，视频拍摄，剪辑，汇报 PPT 制作，项目报告编写，以及未提及模块的编写。

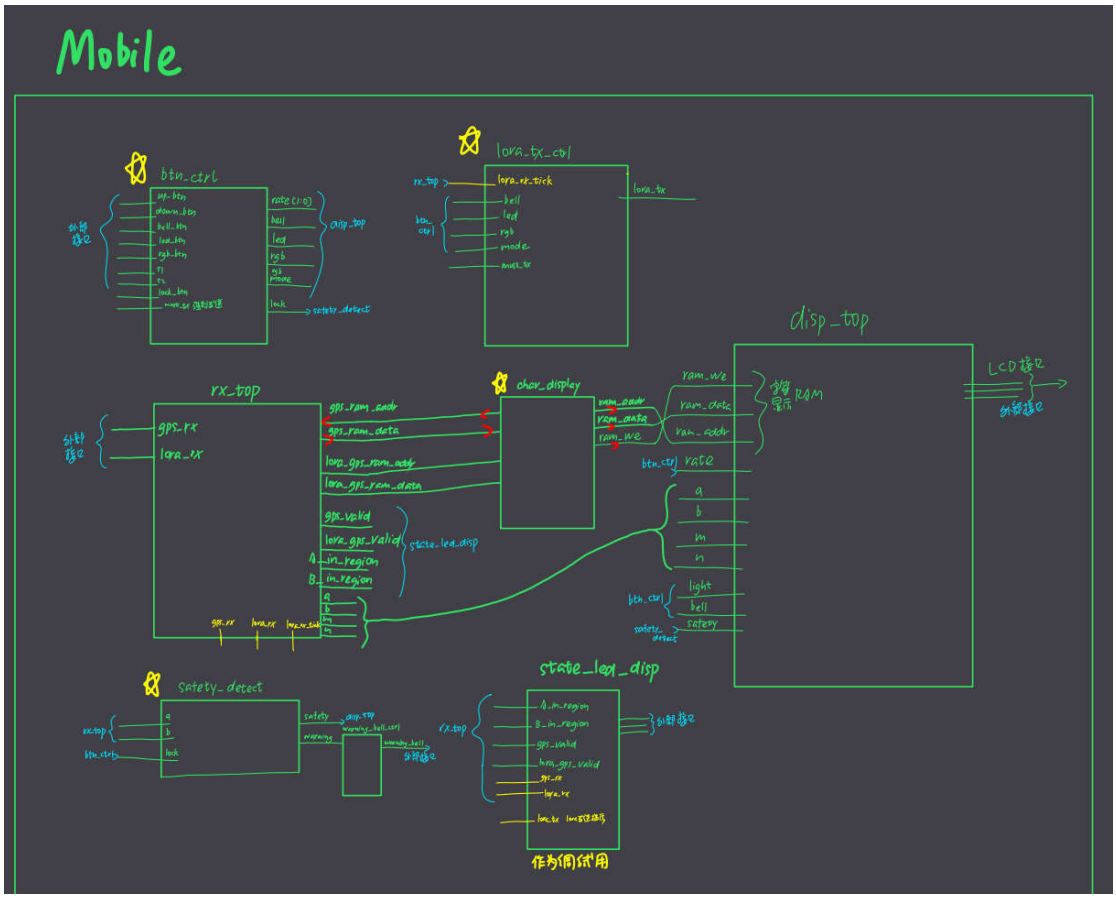
xxx：IIC 驱动，EC11 编码器驱动，OLED 数据手册学习，C 语言字模格式转换，UART 驱动设计，GPS 模块数据手册学习，GPS 数据解读模块设计，程序固化。

xxx：WS2812 驱动设计，灯带动态显示设计，IIC 驱动改进，LORA 无线模块使用学习，地图坐标采集与绘制，视频拍摄。

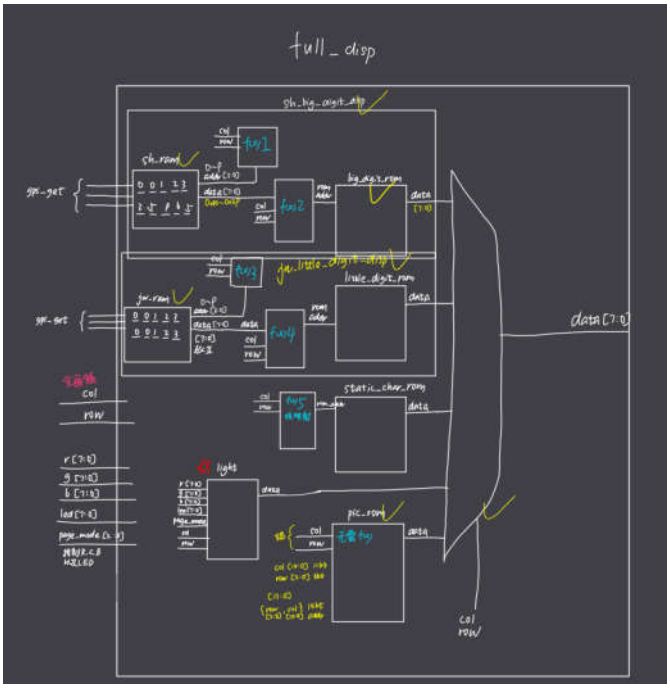
系统整体设计

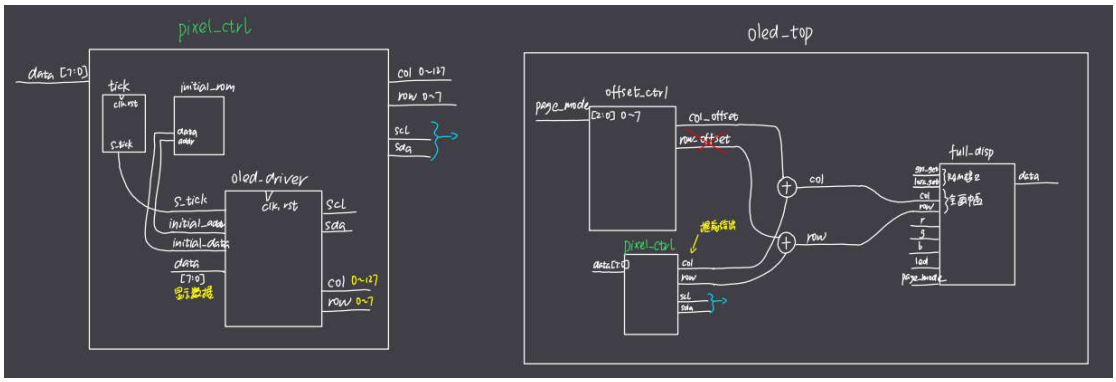
2.1 系统框图

移动端主要模块结构：



自行车端主要模块结构：





2.2 系统方案描述

硬件方案描述

1. 显示屏：

显示屏是整个系统对外信息展示的最重要的模块，由于无论是自行车端还是移动端，都需要极强的便携性，即显示屏要足够小，能够容易的拿在手中或者安装到自行车上，而且功耗也不能太大，因为手持设备的电池容量有限。综上，我们就不能选择像 VGA 显示器那种的大型显示设备，经过查找，我们最终选择了 8080 接口的 3.5 寸 LCD 屏作为移动端的显示屏，使用 IIC 接口的 SSD1306 驱动的 0.96 寸 OLED 屏作为自行车端的显示屏，这两种显示屏的效果确实令人满意，但是由于资料较少，我们在驱动过程中也费了很多精力。

2. 无线通信模块：

由于本项目的自行车端和移动端需要在较远的距离上进行通信，而且教学区楼宇众多，对信号的阻碍较强，所以我们没有使用通讯距离很短的蓝牙，也没有选择信号强度衰减较快的 2.4GHz 通信模块，而是选择了 433MHz 的 LORA 扩频无线模块，经过实际测试，这款模块的距离和发送数据稳定程度还是不错的。

3. GPS 定位模块：

由于自行车端还有移动端都需要获取自身的位置，所以整个项目需要两个 GPS 模块，而且不同模块的定位体系和使用的坐标格式可能不同，所以我们两个端的 GPS 模块使用了一模一样的两个模块，保证了定位精度和一致性，即使在总体坐标偏移的情况下，相对位置依然能够准确显示。

4. 人机交互硬件：

移动端需要控制按钮来控制相应的功能，但是项目后来整个的线序连接较为混乱，使得我们不太容易去直接使用开发板上的按钮，所以我们就外接了几个按钮在包装盒的外面，使得控制更为方便。

自行车端由于需要显示不同的信息，不同的界面之间需要左右切换，而且骑自行车的时候手的移动范围有限，基于此，我们没有使用两个按钮去进行左右切换，而是使用一个 EC11 旋转编码器去达到控制功能，虽然在程序上会复杂一些，但是整体的使用体验提升了很多。

5. 对外提示硬件：

除了显示屏作为主要信息输出途径之外，我们还需要一些调试信息的显示，比如 GPS 定位模块有没有输出数据，GPS 模块输出的数据是不是有效的，以及定位是不是在我们想要的范围之内等信息，这对于初期排查问题是十分有用的，但是实际使用的时候不用显示出来，所以我们直接使用板子上的 LED 作为显示输出。还有就是移动端在报警的时候不光要在 LCD 上显示出来，还要发出报警信号，所以我们使用了一个蜂鸣器来进行报警输出。

6. 其他支持硬件

整个系统需要足够便携，所以需要一个基本的外包装，将内部的线路保护起来，防止线路断开或接触不良，以及减少静电对于板子的影响，最重要的是，拿起来更方便，以及更容易安装到自行车上。我手边有很多纸盒子，而且大小也合适，就拿纸盒子作为整个模块的外壳。

还有就是供电的部分，由于按钮，无限模块，GPS 模块，显示屏等都需要相应的电源，所以需要一个含有较多接口的简易“插排”，所以我们将面包板边上的供电部分取下来作为电源，最主要的问题是不同模块的电源要求不同，例如 LCD 显示屏的背光需要 5V 才能点亮，而有些模块只能工作在 3V3，而且 FPGA 的 GPIO 好像也不建议直接输入 5V 电平，所以我们需要两排供电，分别供给 3V3 和 5V 的电压。

器件选择

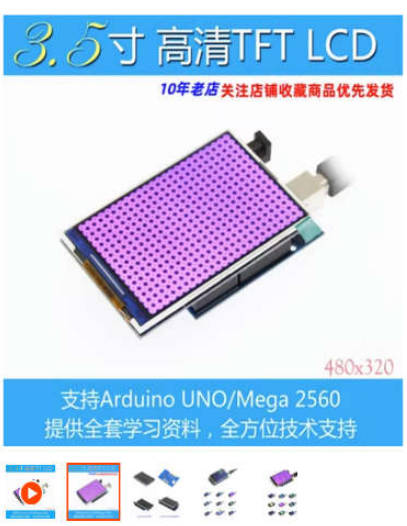
主要器件选型

LLCC68 无线串口模块

ATGM336H GPS 定位模块

SSD1306 IIC 接口显示屏

TFT-LCD 8080 接口显示屏



3.5寸TFT彩屏模块 320X480超高清液晶屏 支持 UNO Mega2560 DUE

价格 ¥38.50 100+ 月销量

淘宝价 ¥38.40 优惠促销

配送 广东深圳 至 浙江杭州西湖区 快递 ¥4.00

现在下单，预计4月17日送达

数量 1 件

立即购买 加入购物车

承诺 7天无理由

WS2812 RGB LED

12位 WS2812 5050 RGB LED 全彩驱动彩灯 圆形开发板

价格 ¥2.60 90 月销量

淘宝价 ¥2.50 新品抢鲜价

配送 广东深圳 至 浙江杭州西湖区 快递 ¥4.00

现在下单，预计6月28日送达

数量 1 件

立即购买 加入购物车

承诺 7天无理由

EC11 旋转编码器

数字旋转编码器模块 转动电位器 旋转电位器 带旋钮帽

价格 ¥3.90 69 月销量

淘宝价 ¥3.80 让利促销

配送

数量 1 件

立即购买 加入购物车

承诺 7天无理由

低电平触发蜂鸣器

电子积木 有源蜂鸣器模块 低电平触发 蜂鸣器控制板

价格 ￥1.10

数量 1 件(库存2件)

立即购买 加入购物车

按钮

新款蓝板 电子积木 大按键模块按钮模块五色套装

价格 ￥3.40

数量 1 件(库存2件)

立即购买 加入购物车

模块设计

移动端主要模块

pixel_map（LCD 字符显示映射模块）

```
module pixel_map(
    input wire [8:0] pixel_x,
    input wire [7:0] pixel_y,
    input wire clk,
    input wire we,
    input wire [9:0] addr,
    input wire [6:0] din,
    output wire pixel_value,
    output wire [15:0] pixel_data
    );
```

引脚说明：

pixel_x	水平坐标输入
pixel_y	垂直坐标输入
clk	系统时钟(100MHz)
we	字符 RAM 写使能
addr	字符 RAM 地址输入
din	字符 RAM 数据输入(ASCII 码低 7 位)
pixel_value	像素二值输出
pixel_data	像素完整输出

模块详细介绍：

这个模块负责字符部分的像素映射，LCD 驱动模块告知此模块当前屏幕刷新到哪里了，此模块根据字符 RAM 中的内容以及传入的显示坐标返回给 LCD 驱动模块此此像素点的颜色信息，由于字符是二值显示，所以如果此像素点有字符，则 pixel_value 为 1，pixel_data 为 0xffff

模块包含的更底层模块：

char_ram font_rom

map_top（地图渲染模块）

```
module map_top(
input wire clk,rst,
input wire [1:0] rate,
input wire [8:0] pixel_x_in,
input wire [8:0] pixel_y_in,
input wire [8:0] b_map_x,b_map_y,
input wire [8:0] a_map_x,a_map_y,
output wire [15:0] disp_data
);
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
rate	地图放大倍数输入
pixel_x_in	水平坐标输入
pixel_y_in	垂直坐标输入
b_map_x,b_map_y	自行车端地图坐标输入
a_map_x,a_map_y	移动端地图坐标输入
disp_data	地图像素信息输出

模块详细介绍：

这个模块也是根据 LCD 驱动模块去给出当前显示像素的像素信息，其中，rate 用于调整放大倍率，00:1 倍 01:2 倍 10:4 倍 11:8 倍，自行车端和移动端的坐标输入为相对于地图原点的坐标，用于在相应位置显示红点和绿点，并且根据移动端位置去调整画面使得移动端永远在地图中心。

模块包含的更底层模块：

pixel2map(倍率映射模块)

map2bit(地图像素信息提取模块)

map_rom(地图信息存储模块)

lcd_driver(LCD 驱动顶层模块)

```
module lcd_driver(
    input wire clk,rst,
    input wire [15:0] data_in,
    output wire [8:0] data_out,
    output wire rst_out,cs,rd,wr,
    output wire [8:0] pixel_x,
    output wire [8:0] pixel_y
    );
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
data_in	当前像素信息输入
data_out	8080 接口数据输出，其中 MSB 作为 RS
rst_out	LCD 复位信号输出
cs	LCD 片选使能
rd	LCD 读取/写入模式控制
wr	LCD 时钟

pixel_x	水平坐标输出
pixel_y	垂直坐标输出

模块详细介绍：

这个模块是直接跟 LCD 相连接的，对于系统内部，输出当前显示像素的坐标信息，输出给字体渲染模块和地图渲染模块，然后渲染模块给出该像素点的像素信息传回到 LCD 驱动模块，构成一个一问一答的信息回路，从而将每个像素点的颜色信息按照 8080 接口的协议格式发送给 LCD 显示屏。这个模块也包含显示屏上电初始化的作用，即每次 FPGA 上电之后首先延时 100ms 对 LCD 进行复位，然后按照协议发送 103 字节的初始化数据，之后便开始循环显示，同时为了更为安全的显示，也就是不会出现像素错位的情况，每刷新一帧需要重新发送设置显示窗口指令，然后再不断传送像素数据，总之，整个底层的显示过程都由这个顶层模块来实现，对于系统内部的接口大幅度简化。

模块包含的更底层模块：

initial_rom:LCD 初始化指令 ROM

background（字符显示区域背景颜色渲染模块）

```
module background(
    input wire font_bit,
    input wire [8:0] pixel_x,
    input wire [8:0] pixel_y,
    input wire led,bell,safety,lock,rgb_on,
    input wire [2:0] rgb_mode,
    output reg [15:0] data
    );
```

引脚介绍：

font_bit	此像素是否为字符
pixel_x	水平坐标输入
pixel_y	垂直坐标输入
led	led 点亮信号（未使用）
bell	bell 响铃信号（未使用）
safety	自行车安全信号
lock	自行车位置锁定信号
rgb_on	RGB 灯光开启信号（未使用）
rgb_mode	RGB 灯光模式信号（未使用）
data	像素颜色信息输出

模块详细介绍：

由于字符显示区域需要不同的背景颜色去显示更多的状态信息，所以在字符渲染模块和 LCD 驱动模块之间增加了此背景渲染模块，默认区域黑底白字，特殊

显示区域例如自行车锁定显示区域在锁定情况下为黑字绿底，在锁定且危险的情况下为黑字红底，由于 RGB 在项目后来没有使用，所以在实际系统中未作连接。该模块无更底层模块。

rx_top（数据接收顶层模块）

```
module rx_top(
input wire clk,rst,
input wire gps_rx,lora_rx,
input wire [5:0] gps_ram_addr_rd,
output wire [7:0] gps_ram_data_rd,
input wire [5:0] lora_gps_ram_addr_rd,
output wire [7:0] lora_gps_ram_data_rd,
output wire gps_valid,
output wire lora_gps_valid,
output wire [8:0] m,n,a,b,
output wire A_in_region,B_in_region,
output wire gps_rx_get,lora_rx_get
);
```

引脚介绍：

clk	系统时钟(100MHz)
rst	系统全局同步复位
gps_rx	gps 模块串口输入
lora_rx	lora 无限模块串口输入
gps_ram_addr_rd	gps 坐标信息 RAM 读取端口（地址）
gps_ram_data_rd	lora 坐标信息 RAM 读取端口（数据）
lora_gps_ram_addr_rd	lora 坐标信息 RAM 读取端口（地址）
lora_gps_ram_data_rd	lora 坐标信息 RAM 读取端口（数据）
gps_valid	gps 模块串口输入信息有效
lora_gps_valid	lora 模块串口输入信息有效
m,n,a,b	自行车，移动端相对地图坐标输出
A_in_region,B_in_region	自行车，移动端在设定区域内
gps_rx_get,lora_rx_get	gps,lora 串口输入有效脉冲输出（未使用）

模块详细介绍：

此模块直接连接到外部的 gps 模块和 lora 无线模块的串口输入，首先对传入的串口信息进行协议上的校对，如果不是想要的数据帧或者是损坏的数据帧则进行舍弃，尽量减少传入系统内部的乱码数量。同时此模块根据协议格式对自行车和移动端的经纬度信息进行串并转换，并通过快速乘法器转换成可计算的数据，并根据坐标范围判断定位坐标是否在预定范围内，最后将可计算的坐标数据进行固定系数的比例缩小以及整体偏移量的叠加最终输出以地图坐为原点的自行车和移动端地图坐标数据，供给地图渲染模块使用。

模块包含的更底层模块：

gps_get（NEMA 协议解读模块与校验模块）

```
module gps_get(
    input wire rx,
    input wire clk,rst,
    output wire [5:0] gps_ram_addr,
    output wire [7:0] gps_ram_data,
    output wire gps_ram_we,
    output wire [3:0] jw_ram_addr,
    output wire [7:0] jw_ram_data,
    output wire jw_ram_we,
    output wire [3:0] sh_ram_addr,
    output wire [7:0] sh_ram_data,
    output wire sh_ram_we,
    output wire gps_valid
    );
```

引脚介绍：

rx	串口信息输入
clk	系统时钟(100MHz)
rst	系统全局同步复位
gps_ram_addr	gps 数据 RAM 写入地址
gps_ram_data	gps 数据 RAM 写入数据
gps_ram_we	gps 数据 RAM 写入使能
jw_ram_addr	经纬度数据 RAM 写入地址
jw_ram_data	经纬度数据 RAM 写入数据
jw_ram_we	经纬度数据 RAM 写入使能
sh_ram_addr	航向速度数据 RAM 写入地址
sh_ram_data	航向速度数据 RAM 写入数据
sh_ram_we	航向速度数据 RAM 写入使能
gps_valid	gps 数据有效信号

模块详细介绍：

此模块是解读和提取串口传入信息的模块，对传入的信息做第一步的校验和处理。

首先在众多数据帧中找到所需要的数据帧头，然后利用不变的逗号数量来定位到具体信息的位置，然后开始接收，如果在过程中接收到了本不该在此位置出现的符号或者固定位数的数据位数发生变化，则判定为信息错误，状态机退回 idle 状态等待下一次接收，对于接收到的有效的数据则在相应的位置按照我们规定的数据 RAM 存储格式存储到 gps 数据 RAM，经纬度数据 RAM 以及航向速度数据 RAM 中，供给后续模块读取使用，存储到 RAM 中的数据，格式简便，也对于不同位数的数据进行了对齐，减少了后面模块的工作量。

模块包含的更底层模块：

uart_tx：串口接收底层模块

mod_m_counter：模 m 计数器，用来作为波特率发生器

gps_ram（gps 数据存储模块）

```
module gps_ram#(
parameter ADDR_WIDTH=6,DATA_WIDTH=8)(
input wire clk,
input wire we,
input wire [ADDR_WIDTH-1:0] addr_a,addr_b,
input wire [DATA_WIDTH-1:0] din_a,
output wire [DATA_WIDTH-1:0] dout_a,dout_b
);
```

引脚介绍：

clk	系统时钟(100MHz)
we	RAM 写入使能
addr_a	A 端口地址
addr_b	B 端口地址
din_a	A 端口写入数据
dout_a	A 端口输出数据
dout_b	B 端口输出数据

模块详细介绍：

这个 RAM 是一个通用的标准 RAM 模版，可以被综合成块 RAM，同时也可以通过修改模块参数来快速改变数据位宽以及存储深度。这是一个双端口 RAM，其中 A 端口可写可读，B 端口只读，在本项目中主要用作数据缓存以及显示 RAM，正是 RAM 的大量使用，使得整个系统的各个部分能在不同的数据速率下工作。

jingwei_process（经纬度处理顶层模块）

```
module jingwei_process(
    input wire clk,rst,
    input wire jw_we,
    input wire [6:0] jw_data,
    output wire [8:0] m,
    output wire [8:0] n,
    output wire in_region
    );
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
jw_we	经纬度串行字符数据输入使能
jw_data	经纬度串行字符数据输入
m	水平地图坐标输出
n	垂直地图坐标输出
in_region	区域内

模块详细介绍：

此模块作为顶层模块，只是对底层模块进行例化和连接

jingwei_ctrl（经纬度缩放及偏移模块）

```
module jingwei_ctrl(
    input wire clk,rst,
    input wire [16:0] jing_num,
    input wire [16:0] wei_num,
    input wire data_en,
    output wire [8:0] wei_m_scaled,
    output wire [8:0] jing_n_scaled,
    output wire in_region
    );
```

引脚介绍：

clk	系统时钟(100MHz)
rst	系统全局同步复位
jing_num	经度数值数据输入
wei_num	纬度数值数据输入
data_en	输入数据有效使能
wei_m_scaled	纬度数据缩放偏移后输出
jing_n_scaled	经度数据缩放偏移后输出
in_region	区域内状态指示

模块详细介绍：

此模块的输入是由乘法器输出并且经过串并转换的经纬度数值数据进行缩放和整体偏移，由于经纬度数据变化范围较大，需要一定比例映射到地图上的像素坐标，同时，也需要一个基本偏移量作为位置矫正。经过实际坐标采集与计算，映射比例确定为除以 32，纬度偏移 18000，经度偏移 7000.

```
localparam [16:0] m_offset = 17'd18000;//纬度偏移(m_offset)
localparam [16:0] n_offset = 17'd7000;//经度偏移(n_offset)

//经度对应n坐标 经度增加n增加
//纬度对应m坐标 纬度增加m增加
//经纬度换算到地图坐标的比例：除以32
//之前采集的经纬度范围：经度:07679-12950 纬度：18762-30110
//简化之后：06000-13000 (07000)   18000-30000 (12000)
//除以32:   218.75 333
```

char2num（组合电路乘法器）

```
module char2num(
    input [34:0] a,
    output [16:0] out
    );
```

引脚介绍：

a	并行字符数据输入
out	转换后数据输出

模块详细介绍：

由于 GPS RAM 内部的坐标信息是 ASCII 码而且是位数分离的，所以要想转换成可以计算的数据需要每一位乘以相应的权重来获得最终数据，但是为了使得系统更可控，我们不去使用乘法器，而是使用基于位运算的纯组合电路去专用于转换数据格式，经过实际测试，其性能符合要求，这个模块在整个系统中发挥了重大的作用，使得地图的坐标显示成为可能。

ser2par（循环移位寄存器）

```
module ser2par(
    input wire clk,rst,
    input wire gps_we,
    input wire [6:0] gps_data,
    output wire data_en,
    output wire [34:0] jing_ch,
    output wire [34:0] wei_ch
    );
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
gps_we	经纬度数据写入使能
gps_data	经纬度串行数据输入
data_en	经纬度数据串并转换完毕信号数据
jing_ch	经度并行数据输出
wei_ch	纬度并行数据输出

模块详细介绍：

由于 GPS 数据解读模块输出的是串行数据，但是 char2num（组合电路乘法器）需要并行数据来进行计算，所以需要一个循环移位寄存器，当 gps_we 有效时，数据移入寄存器，每当一次数据移位完毕的时候，data_en 产生一个时钟周期的高电平，表示移位结束，同时后面的模块根据此结束信号进行数据锁存和并行计算，确保数据是完整正确的。

lora 部分由于数据格式相近，直接使用 gps 部分的模块进行复用，结构也是相似的，这里不再说明。

btn_ctrl（按钮控制模块）

```
module btn_ctrl(
    input wire clk,rst,
    input wire up_btn,down_btn,
    input wire bell_btn,led_btn,rgb_btn,lock_btn,
    input wire s1,s2,
    output wire [1:0] rate,
    output wire bell,led,rgb,lock,
    output wire [2:0] rgb_mode
    );
```

引脚介绍：

clk	系统时钟(100MHz)
rst	系统全局同步复位
*_btn	各个按钮的原始输入
s1	编码器 s1 原始输入
s2	编码器 s2 原始输入
rate	放大倍率控制输出
bell, led, rgb, lock	各个控制信号输出
rgb_mode	RGB 模式输出（未使用）

此模块包含对于按钮输入的去抖，常开常闭的控制，以及编码器的消抖和旋转方向判断，并且输出相应的状态信号供给后级模块使用。

safety_detect（安全检测模块）

```
module safety_detect(
    input wire clk,rst,
    input [8:0] a,b,
    input wire lock,
    output reg safety,//Safe:0  Unsafe:1
    output wire warning_bell
    );
```

引脚介绍：

clk	系统时钟(100MHz)
rst	系统全局同步复位
a,b	自行车端地图坐标输入
lock	锁定状态输入
safety	安全状态输出
warning_bell	报警蜂鸣器输出

模块详细介绍：

此模块负责自行车位置的安全监测，并且直接控制蜂鸣器，并向系统内部输入安全状态信号。模块不断接收输入的自行车坐标，当检测到 lock 上升沿也就是开启一次锁定时，模块会锁存当前自行车坐标，然后不断监测之后输入进来的

自行车坐标，当 x,y 任意方向的差值超过安全阈值时，模块立即报警，并且输出不安全信号，直至 lock 变为低电平即解除锁定时停止报警，并准备下一轮的锁定。

以上就是移动端的主要模块介绍，还有些模块最终没有投入使用或者是底层模块没有进行介绍。

自行车端主要模块

oled_top（OLED 显示驱动顶层模块）

```
module oled_top(
    input wire clk,rst,
    input wire sh_ram_we,
    input wire [3:0] sh_ram_addr,
    input wire [7:0] sh_ram_data,
    input wire l_tick,r_tick,
    input wire jw_ram_we,
    input wire [3:0] jw_ram_addr,
    input wire [7:0] jw_ram_data,
    input wire [7:0] r,g,b,led,
    output wire scl,sda
    );
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
sh_ram_we	航向速度数据 RAM 写入使能
sh_ram_addr	航向速度数据 RAM 写入地址
sh_ram_data	航向速度数据 RAM 写入数据
l_tick,r_tick	旋转编码器左右旋转信号
jw_ram_we	经纬度数据 RAM 写入使能
jw_ram_addr	经纬度数据 RAM 写入地址
jw_ram_data	经纬度数据 RAM 写入数据
r,g,b,led	灯光信号（未使用）
scl	IIC 时钟信号
sda	IIC 数据信号

模块详细介绍：

这个模块是自行车端显示部分的最顶层模块，GPS 数据解读模块将解读后的数据传入到本模块中的 RAM 进行存储，本模块根据旋转编码器的输入来改变显示内容，并不断刷新显示屏。

full_disp（全画幅显示模块）

```
module full_disp(
    input wire clk,rst,
    input wire sh_ram_we,
    input wire [3:0] sh_ram_addr,
    input wire [7:0] sh_ram_data,
    input wire jw_ram_we,
    input wire [3:0] jw_ram_addr,
    input wire [7:0] jw_ram_data,
    input wire [10:0] col_all,
    input wire [2:0] row_all,
    input wire [7:0] r,g,b,led,
    output reg [7:0] data
    );
```

此模块跟上面的顶层模块端口基本相同，不一样的是该模块传入的是全画幅坐标，由于整个显示画面是一个大长条，而 OLED 只是显示其中的一部分，但是对于 FPGA 是拿一个长条去渲染的，所以内部需要对全画幅进行不断刷新。

sh_big_digit_disp（速度航向大数字显示模块）

```
module sh_big_digit_disp(
    input wire [10:0] col_all,
    input wire [2:0] row_all,
    input wire clk,
    input wire sh_ram_we,
    input wire [3:0] sh_ram_addr,
    input wire [7:0] sh_ram_data,
    output wire [7:0] sh_big_digit_data
    );
```

col_all	全画幅列地址输入
row_all	全画幅行地址输入
clk	系统时钟(100MHz)
sh_ram_we	航向速度数据 RAM 写入使能
sh_ram_addr	航向速度数据 RAM 写入地址
sh_ram_data	航向速度数据 RAM 写入数据
sh_big_digit_data	大数字显示像素数据输出

模块详细介绍：

此模块是渲染大数字的顶层模块。该模块根据传入的全画幅坐标和当前 RAM 中存储的数据来给出当前 OLED 显示的像素列的显示数据，从而将大数字显示出来。

此模块包含的底层模块；

sh_ram（速度航向数据 RAM）

function1（字符显示映射函数 1）

function2（字符显示映射函数 2）
div24（余 24 计算器）
big_digit_rom（大数字字模 ROM）

pic_rom（图片 ROM）

```
module pic_rom(
    input wire clk,
    input wire [13:0] addr,
    output reg [7:0] data
    );
```

引脚说明：

clk	系统时钟(100MHz)
addr	图片 ROM 地址
data	图片数据输出

模块详细介绍：

自行车端需要显示多张图片，此模块中存放了 5 张，最终使用 3 张图片。

前面的坐标映射模块将当前显示坐标转换成能够在 ROM 中寻址的坐标，然后输出像素数据到 OLED 显示底层模块。

jw_little_digit_disp（经纬度小数字显示模块）

```
module jw_little_digit_disp(
    input wire [10:0] col_all,
    input wire [2:0] row_all,
    input wire clk,
    input wire jw_ram_we,
    input wire [3:0] jw_ram_addr,
    input wire [7:0] jw_ram_data,
    output wire [7:0] jw_little_digit_data
    );
```

小数字的显示原理和方法与大数字基本一致。

此模块包含的底层模块：

jw_ram（经纬度信息 RAM）

function3（字符显示映射函数 3）

function4（字符显示映射函数 4）

little_digit_rom（小数字字模 ROM）

data_ctrl（IIC 数据传输控制模块）

```
module data_ctrl(
    input wire clk,rst,
    input wire [7:0] rom_data,
    output wire [5:0] rom_addr,
    input wire [7:0] data_in,
    output wire [6:0] col,
    output wire [2:0] row,
    input wire iic_done,
    output wire [7:0] iic_data,
    output wire dc,
    output wire iic_start
    );
```

引脚说明：

clk	系统时钟(100MHz)
rst	系统全局同步复位
rom_data	SSD1306 初始化指令 ROM 接口（数据）
rom_addr	SSD1306 初始化指令 ROM 接口（地址）
data_in	显示像素输入
col	显示列地址输出
row	显示行地址输出
iic_done	IIC 单次传输完成
iic_data	IIC 单次传输数据输出
dc	IIC 数据/指令选择
iic_start	IIC 单次传输启动

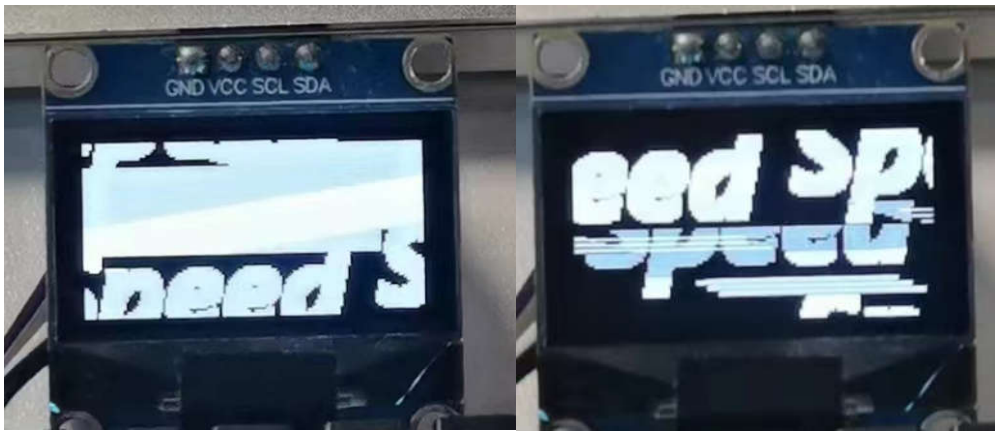
模块详细介绍：

此模块控制着整个 IIC 传输的内容，也操控着 IIC 底层模块，上电之后此模块首先将初始化指令 ROM 中的数据供给 IIC 底层模块发送到 SSD1306 使其初始化，之后便开始不断输出当前显示的行列地址，同时锁存传入的像素坐标信息，并发送到 SSD1306，类似 LCD，每刷新一帧，都要重新设置显示窗口，这也是此模块自动完成的，减少了后续模块的工作量。

每次设置窗口的重要性：（OLED 显示屏接触鲁棒性测试）

1. 每次上电只初始化一次，之后持续输入数据：

可见，当 SCL 线或者 SDA 线接触不良时，后续所有的显示图形都发生了偏移。



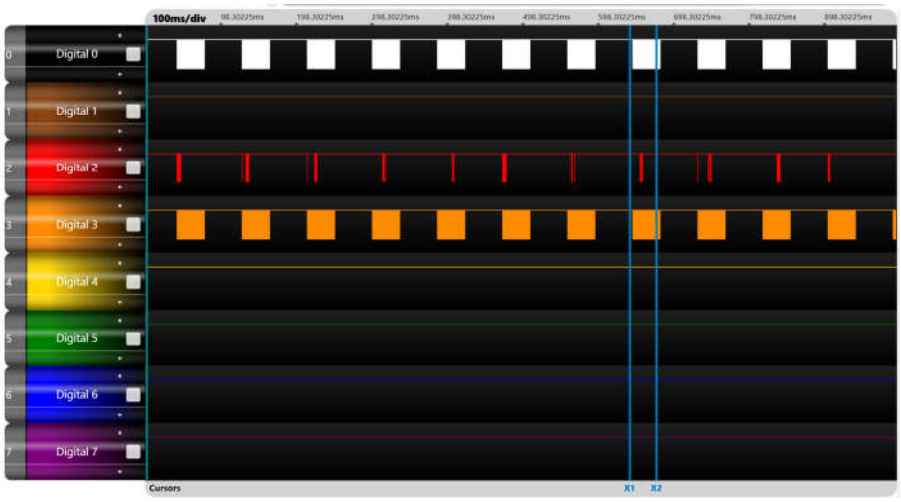
2. 每次刷新一帧后都重新设置窗口

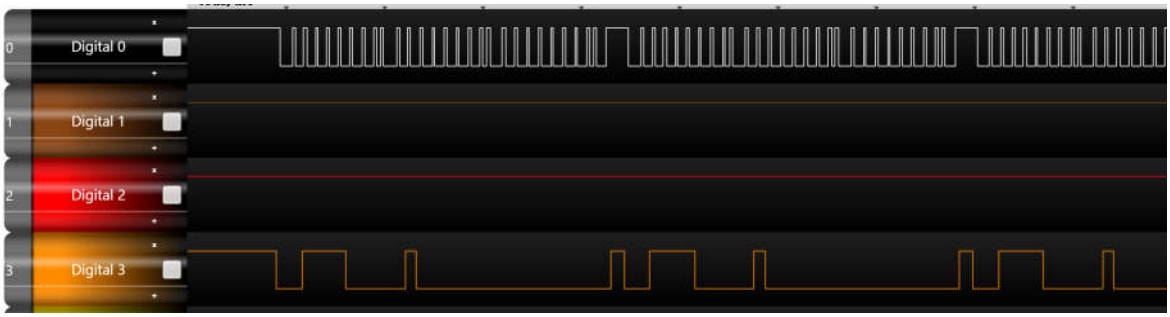
可见，即使 SCL,SDA 线频繁通断，图像都能在下一帧恢复正常。

系统模块实验与仿真

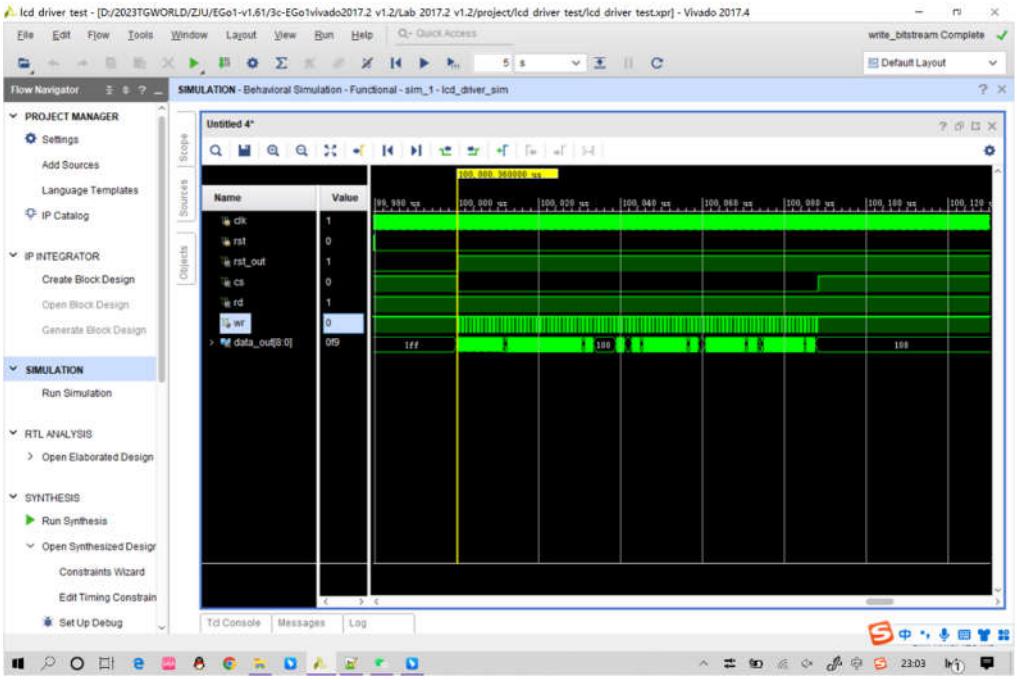
本项目过程中遇到了很多的问题，我们也进行过无数次仿真以及实际逻辑分析仪的分析，下面对一些有记录的实验过程进行展示：

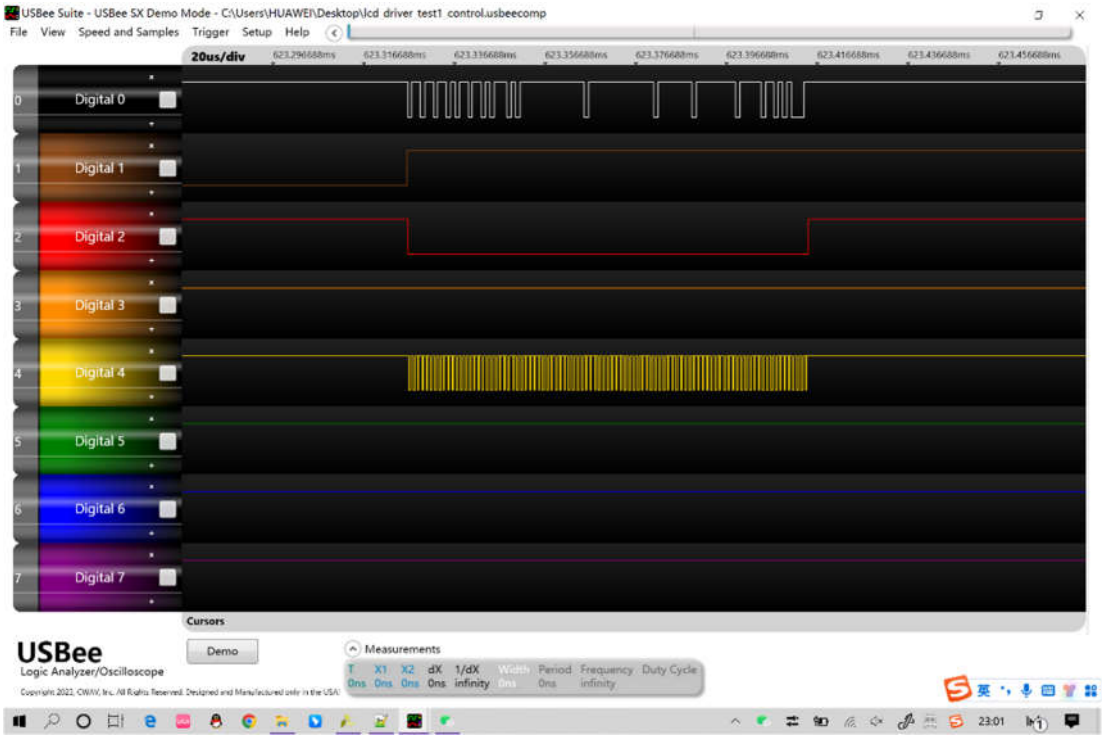
OLED 横条滚动测试逻辑分析仪捕获数据：



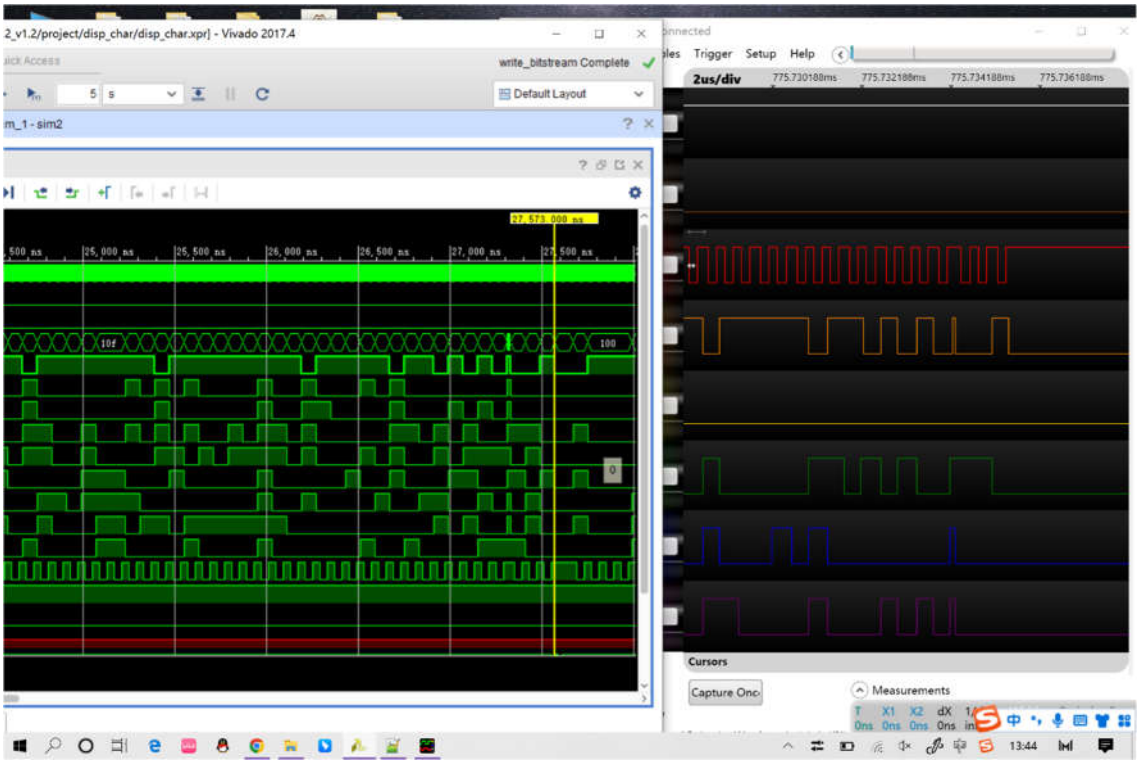


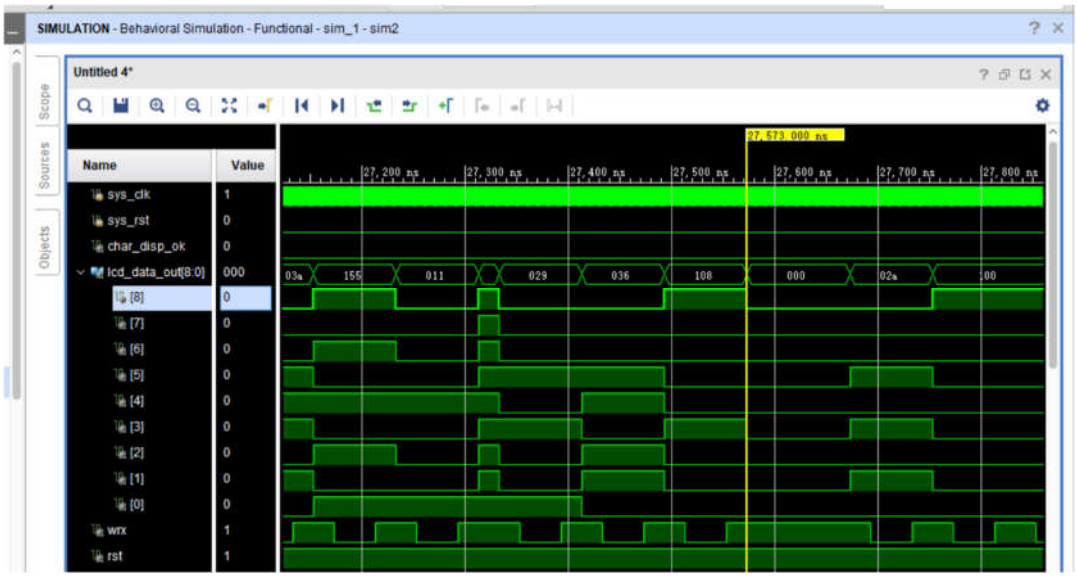
采用 ASMD 设计方法的 LCDdriver-V1 仿真与实际输出比较测试：





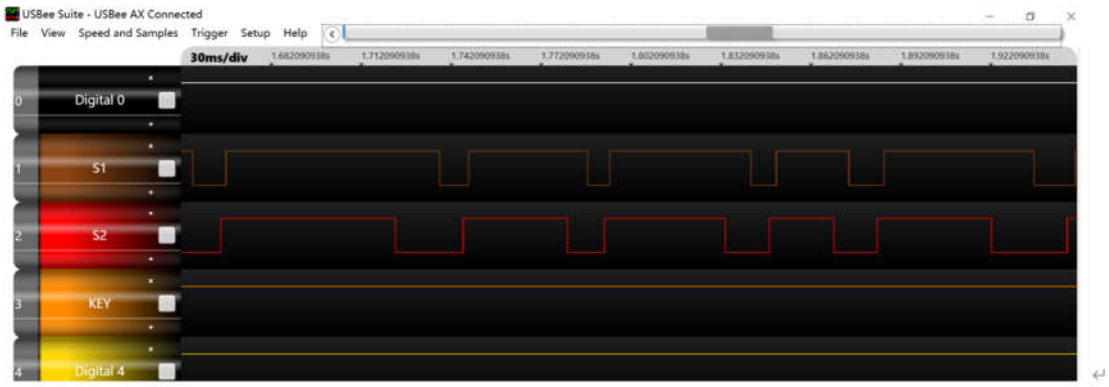
早期LCD驱动（仿真与实际不一致+状态机卡死） 仿真与实际输出对比实验



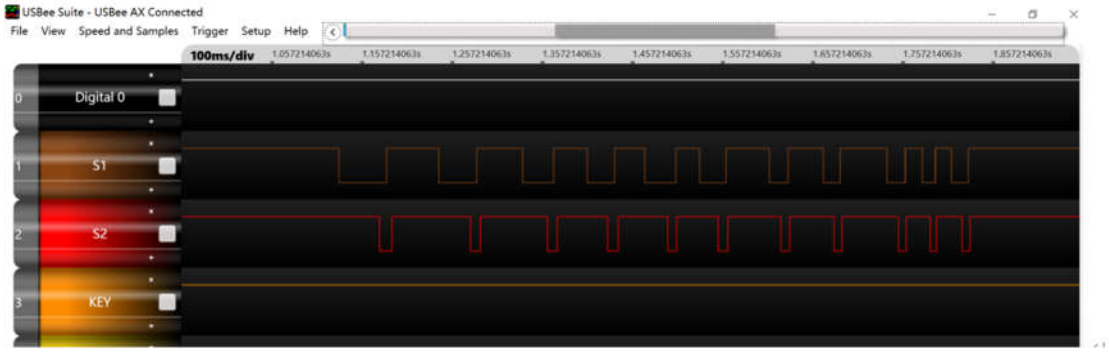


EC11 编码器驱动电平特性测试

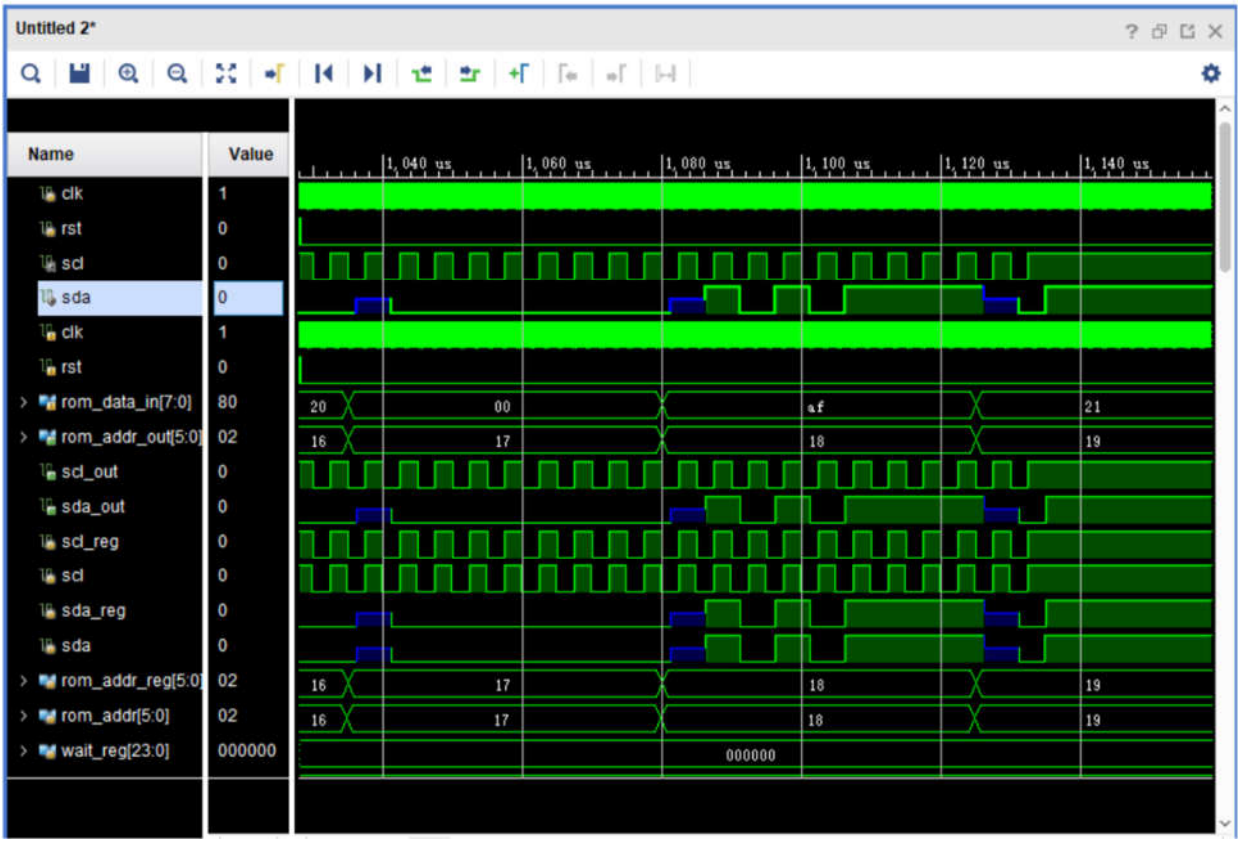
顺时针:

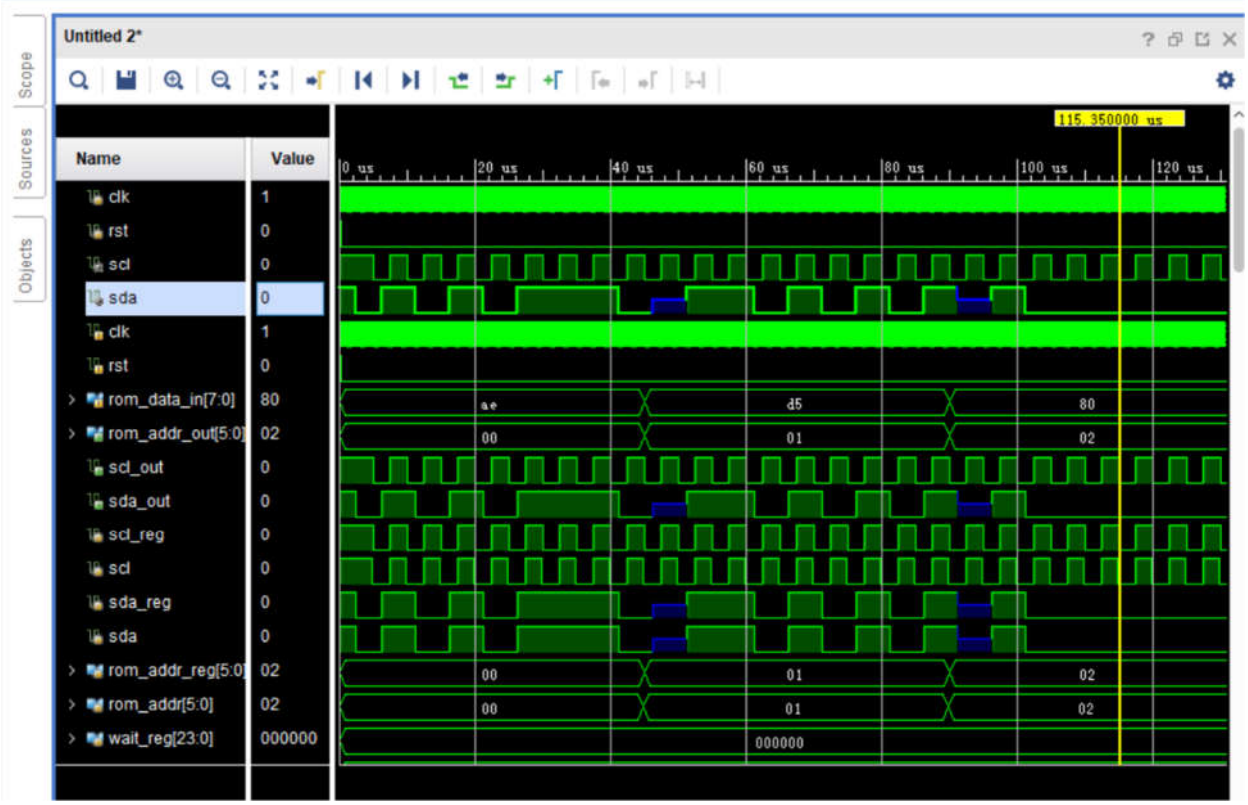


逆时针:



IIC驱动设计过程中数据格式仿真

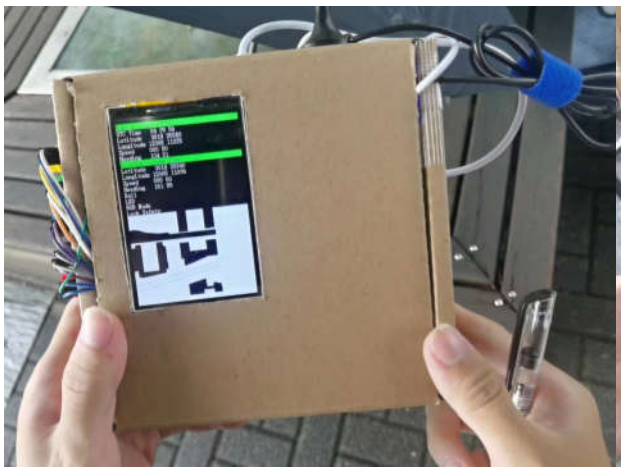


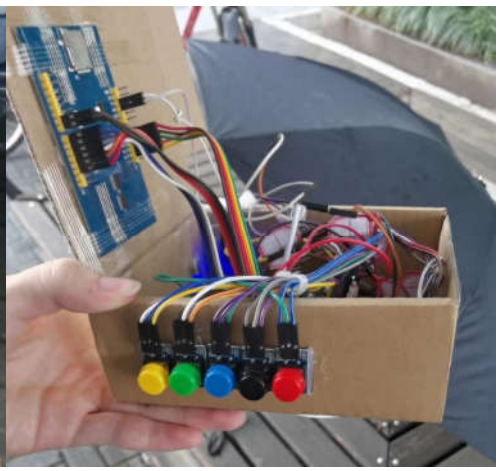


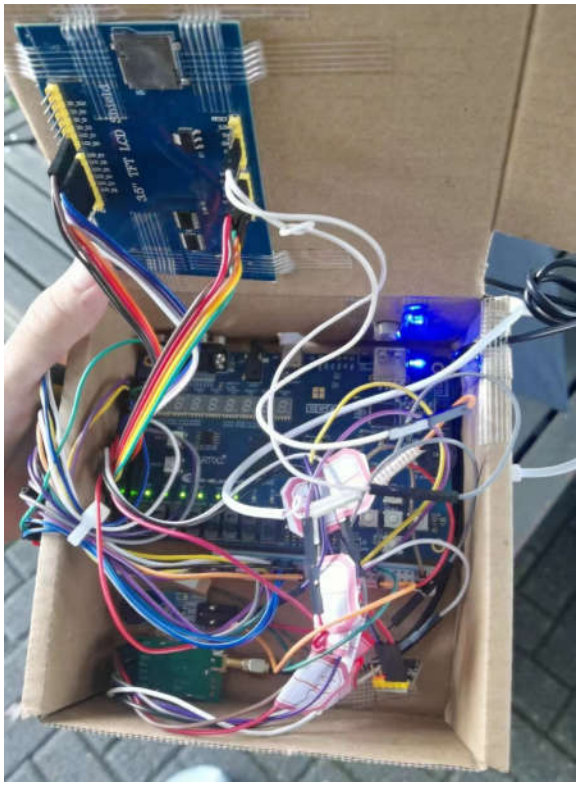
课程设计总结

作品展示图

移动端







自行车端

经验与教训

在整个项目开展的过程中，我们积累了很多开发经验：

1. 对模块的连接关系进行系统记录与显示

项目开展初期，模块很少，并且大都是模块单独开发，单独测试，所以并没有出现模块混乱的情况，但是项目后期所有模块整合起来进行系统联调的时候问题就显现出来了，比如突然想在某个环节插入一个模块，那么就应该清楚这个环节的前一个模块和后一个模块的关系，以及此节点对于整个系统的影响，而如果只观察文件名或者文件框架,是很难去判断好到底往哪里添加的,于是我们当时先暂停开发，先整理好目前的项目总体框架图，梳理好之后再继续调整和添加模块，虽然当时花费了额外的一部分时间去梳理，但是总体上减少了后续排查问题的时间。所以，以后不论项目大小，对于模块之间的连接关系以及包含关系从头开始就要有一个合理的记录体系，方能在查找问题或者修改模块的时候更为方便。

2. 对于位宽要格外注意

我们在项目中也被位宽问题困扰过很多次。例如，对于串口接收的数据是 8 位的，但是 ASCII 码可显示的字符只用到 7 位，由于字体 ROM 需要根据 ASCII 进行寻址，为了节约地址空间，我们使用串口数据的[6:0]进行寻址，这也导致了字体显示模块的 ASCII 输入是和串口接收到的位宽不同，虽然位宽只有单向减少的过程，并没有对于整个系统造成影响，但是我们认为还有不规范的地方，所以，对于位宽需要特殊变化的模块，首先应在模块调用的时候写好注释，并最好在模块内部进行位宽的裁剪操作，而不是在模块例化的时候进行裁剪输入。

3. 线网命名规范

众多模块之间需要很多 wire 的连接，而且很多 wire 的性质和功能是相同的但是连接目标不同，例如字体 ROM 的 addr 线以及 data 线，还有字体 RAM 的 we,addr,data 线，很多线必然会造成命名上的冲突，我们之前出现过不小心将模块内部连线连接到输入输出线上，产生的问题也很难去排查，所以，之后我们对于内部连线都使用 wire_前缀，尽量减少此低级错误的产生。

4. 模块化设计与检测

由于基本所有模块都是我们从头开始写的，并且很多东西和写法都是我们认为可行，但是实际不一定行的东西，所以我们只要每写完一个新的模块，就对其进行

全面的测试，包括仿真，实际上板子等测试方式，因为如果这个模块存在不可行的地方或者一些不太显眼的 bug，那么以后系统整体出现问题也就十分难以排查了，毕竟 FPGA 的无论是多次实际验证还是硬件调试都是很费时间和精力的,所以我们对于每个模块都认真检查，确保每个环节不会出现问题，虽然这有些时候十分枯燥，但还是胜过于后期痛苦地排查寻找问题的根源。

5. 合理利用多种工具去排查问题

在项目初期驱动显示屏的时候，往往出现显示屏无法正确读取指令的情况或者刷新错误，我们最初根本不知道问题出在哪里，而且淘宝商家给出的资料都是 MCU 方面的，根本没有 FPGA 的资料，而且我们也基本没有找到相关移植到 FPGA 上并且成功驱动的示例，所以我们只能参考 STM32 的例程去查明问题根源。我们通过将 STM32 的例程的函数嵌套全部展开，展开到最低的单次传输，并通过修改程序，将 STM32 模拟成一个 FPGA 去进行调试，利用 STM32 编译下载速度快的优势，我们做了很多实验，通过不断修改参数去获得不同的显示效果，例如显示屏翻转指令，地址逆映射指令，显示屏刷新率指令，像素寻址指令等，都是通过不断的查阅数据手册并相应去测试不同指令后才达到的较好的显示效果，然后我们将获得的指令记录下来并打进相应的 verilog 里去初始化 ROM，而不是直接在 FPGA 上测试不同指令，那样的话调试时间就无法估量了。

逻辑分析仪在整个项目中也帮了大忙。很多时候硬件可能并不会按照你想要的形式去产生电平信号，比如对于 STM32，可能你的 GPIO 相应的时钟没有开启，或者有时候 AFIO 的时钟忘开了,那么实际显示屏如果没有点亮的话你就可能开始怀疑时序上的问题,到最后发现引脚压根就没有发出信号,那么是会很崩溃的。对于 FPGA，则可能是内部连线并没有引出去,导致引脚没有信号。所以,每当遇到问题的时候，不妨先拿逻辑分析仪测试一下，能发现很多容易忽略的问题，我们利用逻辑分析仪也发现了时钟极性的问题，以及 RS,RD 引脚等的电平配置问题，帮助我们最终成功写出了 LCD 通用驱动模块。

6. 模块复用设计

通用设计，在大型项目中也是十分重要的。例如 LCD 驱动模块设计初期，我们只设计了屏幕上半部分即显示字符的部分，所以其显示范围的指令以及整体的显示映射结构都是基于那部分的字符而设计的，所以到后来我们显示地图的时候就感觉十分不好受了，基本上得修改模块的大部分，以及相应的指令参数，改起来也是比

较费劲。所以经过这个模块的经验，我们在后来设计 GPS 数据解读模块的时候就考虑了其通用性，使得面对自行车端和移动端不同的数据种类以及不同格式需求，已经预留了相应的接口，对于不同的端选择连接即可，而不是对于每个端单独设计一个相应的模块。还有比如模 m 计数器的通用设计，UART 的 TX,RX 模块等，经过模块化设计之后直接拿过来就能用，在项目后期节约了很多时间。

但是，也出现了一部分问题，例如模 m 计数器在例化的时候有些时候会因为懒忘记修改参数，由于 UART RX 模块需要计数器来工作，所以参数的错误使其无法正常接收数据，而且这种错误也是及其隐蔽，就是因为参数还是之前一个高频模块所使用的参数，所以对于每个通用的模块，在每个例化的时候都要有写全模块参数的习惯，即使很多模块参数都是一致的。

回顾展望

这个项目或许是上大学以来我们面对的最为困难，压力最大的项目了，在立项的时候，我们并没有意识到这个项目实际做起来是这样的复杂与困难，并充满着如此多的不确定性。我之前有过 STM32 的开发经验，但没有接触过 FPGA，以为 FPGA 这个东西应该也不会差太多吧，但事实证明我大错特错了，我也预料到会遇到很多问题，但是也没预料到会有这么多问题。事实上，FPGA 需要你去进行最为底层的操作，这比 MCU 去写一行函数要复杂的多，这也意味着只要实现一个 MCU 看来很容易的操作就需要大量的工作，而且 verilog 也不是我们刚开始以为的 verilog,虽然最初我们是知道它是不同于 c 语言之类的，它是有很多规范的，c 语言的不同写法最后实现出来是一样的，但是 verilog 不同的写法最后综合出来的实际电路却可能千差万别，或者说是不容易控制的，所以我们更加在意 verilog 的稳定写法，确保不会综合出什么奇怪的东西去使得整个系统变的不可控。总体想说的就是对于自己没有完全了解的东西不要摊大饼，否则只会被现实的压力推着痛苦地前进，但是现在看来，我们并没有对于当初的立项而感到后悔，即使这个项目花费了我们很多个人时间,因为整个过程中,我们积累的无数经验,我们对于 FPGA 更为深入的了解，我们合理面对压力的态度，还有处理实际问题的方法论都有了很大的进步，这相比与我们付出的时间与精力，以及熬的那些夜来说都是十分值得的。

整个项目的过程中我认为收获最大的就是我们成功做到了独立去开发一个完整的项目。当然，这并不是我们最初的想法或者锻炼目标，我们当初也是在网上查找了很多资料，也去找各种开源的示例，但是，我们使用的显示屏，给我们的感觉就是基本没有

人拿 FPGA 去驱动过这种显示屏，我们大多数找到的都是 VGA 的资料，我们又找了正点原子以及野火等较为热门的开源资料，但他们的 TFT-LCD 是 RGB 接口的，RGB 接口也是跟 VGA 很接近的，我们也没法移植到 8080 接口。总之就是在项目初期我们点亮一个显示屏就十分的困难，助教也曾建议转向 VGA 等别人使用过的模块，而不是卡在这里去解决不知何时能解决的问题。那一周或许是最为纠结的一段时间了，于是我就去图书馆查找相关资料，幸运的是，在一本书中，我遇到了 ASMD 图，我遇到了基于 RT 方法学的设计方法。我刚开始对这种方法持怀疑态度，但尝试着使用这种方法去设计一个去抖电路作为测试之后，我意识到了这种方法的优点和极强的通用性。于是我基于此设计出了通用LCD驱动模块,没想到竟然跟整个系统配合的如此完美,并且在时序上也很容易控制，于是我又写出了数据转移模块以及定时发送模块等，也没有任何问题，所以我将这种方法推荐给组员，后续所有的模块都是基于此方法完成的，当然，这种方法是目前我所认为的最好的开发方式，但是受限于视野与见识，这可能有很大的局限性，所以在未来的开发中我会继续探索更好的方法。总之，这个方法对于我们的项目的完成起了重大的作用，同时，我们也真正意识到了独立开发的优势，我们可以随时对任何模块的任何地方做出调整与修改，可以实现任何理论上可行的效果，而不用受开源模块的各种时序层面或者位宽等层面的制约，也可以去掉不必要的模块部分使得整个项目工作起来更为简洁和高效，总之自由度是很高的。而且，由于是独立开发，所有的模块设计方法一致，码风一致，写法一致，使得整个系统也是配合的很好。整个过程中，我们也最充分调动了我们的思维去解决实际问题，并基于自己不太宽广的见识去尽力创造出一个能用的模块，来达到最终的目的，总之，知识与视野是一方面，如何充分调用和创造，也是重要的一方面。

对于未来，我们会以更谦虚的态度，更为合理的方法论去完成各种实际项目，去解决各种实际问题，在不断学习成长中探索世界，改造世界，为中国，为世界的发展做出我们的一份贡献。