浙江大学
ZHEJIANG UNIVERSITY

多功能自行车助手项目

《数字系统》课程作业结题报告

日期：2023.4.25-2023.6.19

视频演示

设计过程遇到的最大问题的解决：使用全局时钟网络驱动所有模块
即所有的always模块都直接由100MHz的sys_clk驱动，而不是使用分频后的时钟
这种方法解决了之前仿真与实际不一致以及各种时序上的问题
分频使用tick来实现，不仅时钟信号更为稳定，而且可以实现奇数分频

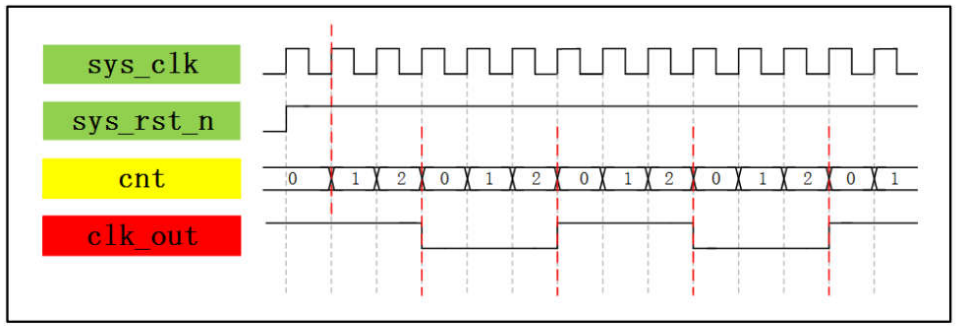


图 17-3 六分频波形图

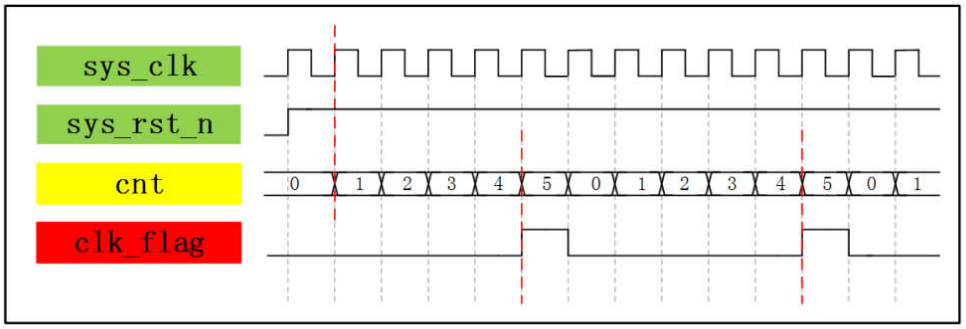
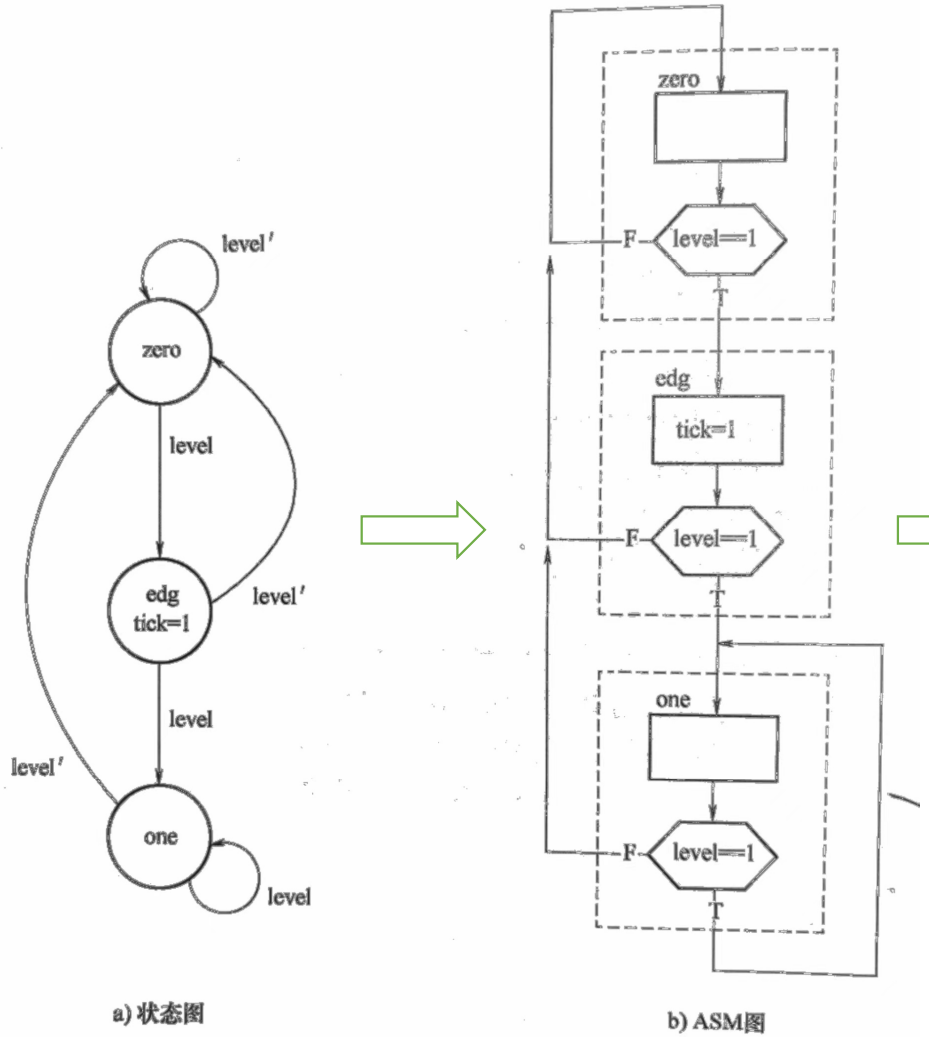


图 17-4 六分频降频方法波形图

```
case(state_reg)
    st_idle:begin
        scl_next=1'b1;
        sda_next=1'b1;
        if(iic_start)begin
            data_next=din;
            sladdr_next=SLAVE_ADDR;
            if(dc)begin
                ctrlbyte_next=8'h40;
            end
            else begin
                ctrlbyte_next=8'h00;
            end
            d_next=0;
            state_next=st_start1;
        end
    end
    st_start1:begin
        if(s_tick)begin
            state_next=st_start2;
            sda_next=1'b0;
        end
    end
    st_start2:begin
        if(s_tick)begin
            state_next=st_start3;
        end
    end
    st_start3:begin
        if(s_tick)begin
            state_next=st_start4;
            scl_next=1'b0;
        end
    end
```

设计方法的改进——使用ASM&ASMD（算法状态机图）进行开发



a) 状态图　　b) ASM图

verilog

优点：

1.虽然要多画ASM图（甚至有些图还很大），但是总体的代码编写效率大幅提升，因为ASM图可以直接翻译成Verilog。

2.不易出错，使用ASM图后，对于一个全新的模块，从头编写平均一次综合即可完成验证，更不会出现时序上错误，无需将时间过多地花费到仿真和综合实现上。

3.方法通用性强，使用ASM帮助我们独立完成了后续所有模块的开发，而不用受开源驱动的种种限制，也得以通过重写改进了LCD驱动和IIC驱动，能够更适配到我们的项目之中。

状态机写法的改进

采用寄存器与组合电路完全分离的写法
以我们的个人观点来看是要比三段式状态机更为合理和稳定（综合稳定性）的

特点：1.每一个寄存器（变量）都有一个_reg和_next

```
reg [1:0] state_reg,state_next;
reg [3:0] s_reg,s_next;//对采样点进行计数
reg [2:0] n_reg,n_next;//对数据位进行计数
reg [7:0] b_reg,b_next;//接收数据缓冲器
```

2.整个模块由两个always块组成，其中一个由clk驱动，另一个为纯组合电路

```
//FSMD
always@(posedge clk or posedge reset)begin
    if(reset)begin
        state_reg<=idle;
        s_reg<=0;
        n_reg<=0;
        b_reg<=0;
    end
    else begin
        state_reg<=state_next;
        s_reg<=s_next;
        n_reg<=n_next;
        b_reg<=b_next;
    end
end
```

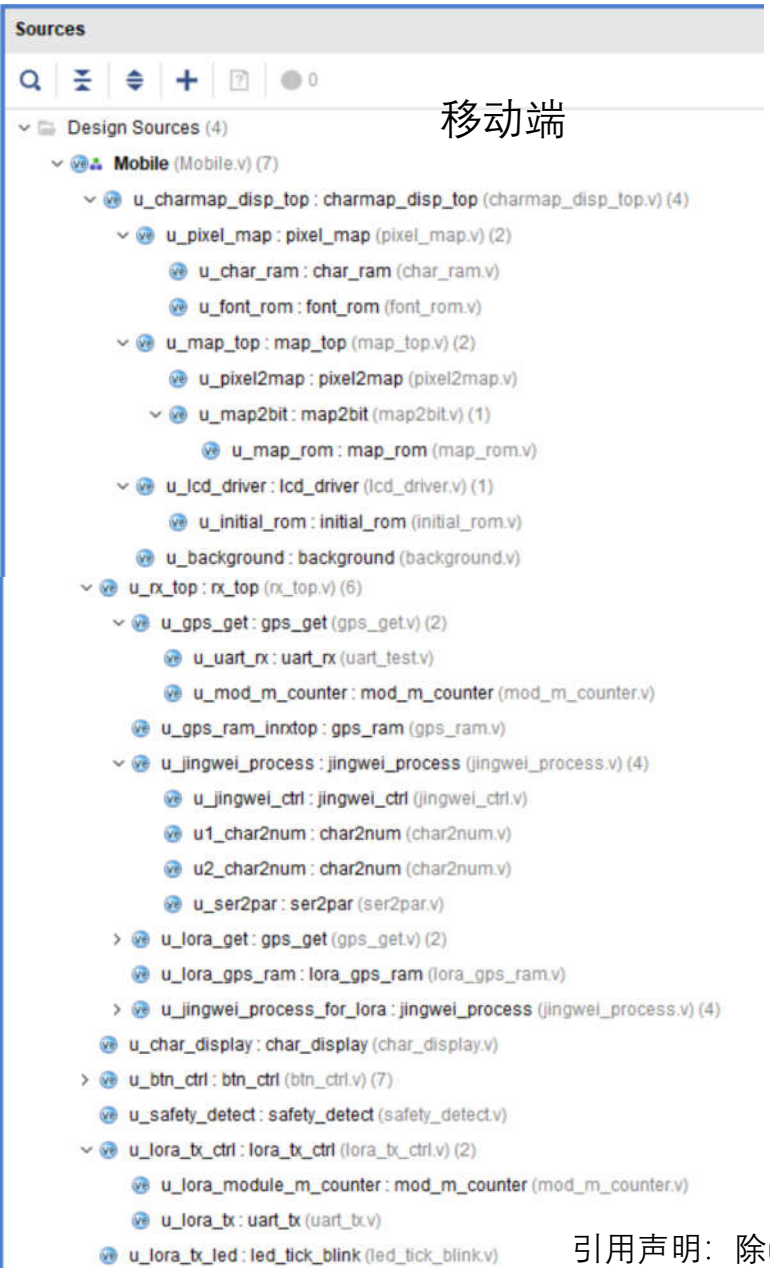
3.组合电路中变量赋初值，之后按需改变默认值，避免了latch的生成

4. <=只出现在clk块中，减少了很多不该出现的错误

```
//FSMD组合逻辑
always@(*)begin
    state_next=state_reg;
    s_next=s_reg;
    n_next=n_reg;
    b_next=b_reg;
    rx_done_tick=1'b0;

    case(state_reg)
        idle:begin
            if(~rx)begin
                state_next=start;
                s_next=0;
            end
        end
        start:begin
            if(s_tick)begin
                if(s_reg==7)begin
                    state_next=data;
                    s_next=0;
                    n_next=0;
                end
                else begin
                    s_next=s_reg+1;
                end
            end
        end
```

程序结构

移动端



自行车端

引用声明：除uart_rx,module_m_counter,其他模块均为原创

参考资料

BDS/GNSS 全星座定位导航模块

ATGM336H-5N

用户手册

杭州中科微电子有限公司
杭州市滨江区江南大道 3850 号创新大厦 10 楼
电话：0571-28918107
传真：0571-28918122
网站：http://www.icofchina.com

WS2812B-V5/W

智能外控集成 LED 光源

主要特点

- 灰控制电路与LED点光源共用一个电源。
- 控制电路与RGB芯片集成在一个5050封装的元器件中，构成一个完整的外控像素点。
- 内置信号整形电路，任何一个像素点收到信号后经过波形整形再输出，保证线路波形畸变不会累加。
- 内置上电复位和掉电复位电路。
- 每个像素点的三基色颜色可实现256级亮度显示，完成16777216种颜色的全真色彩显示。
- 端口扫描频率2KHz/s。
- 串行级联接口，能通过一根信号线完成数据的接收与解码。
- 当刷新速率30帧/秒时，级联数不小于1024点。
- 数据发送速度可达800Kbps。
- 光的颜色高度一致，性价比高。
- 具有电源反接不会损坏。
- 外部不需要包含电容在内的所有任何电子元器件。

主要应用领域

- 消费性电子产品领域。
- LED灯饰亮化领域。
- 电脑及周边设备(游戏设备)各种电器设备领域。

产品概述

WS2812B-V5是一个集控制电路与发光电路于一体的智能外控LED光源。其外型与一个5050LED灯珠相同，每个元件即为一个像素点。像素点内部包含了智能数字接口数据锁存信号整形放大驱动电路，还包含有高精度的内部振荡器和可编程定电流控制部分，有效保证了像素点光的颜色高度一致。

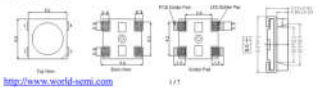
数据协议采用单线归零码的通讯方式，像素点在上电复位以后，DIN端接受从控制器传输过来的数据，首先送过来的24bit数据被第一个像素点提取后，送到像素点内部的数据锁存器，剩余的数据经过内部整形处理电路整形放大后通过DOUT口开始转发输出给下一个级联的像素点，每经过一个像素点的传输，信号减少24bit。像素点采用自动整形转发技术，使得该像素点的级联个数不受信号传送的影响，仅受刷新信号传输速度要求。

高达2KHz的端口扫描频率，在高清视频大的播报下都不会出现闪烁现象，非常适合高清移动产品的使用。

50μs以上的**RESET**时间，达到中断也不会引起误复位，可以支持更低频率，价格便宜的MCU。

LED具有低电压驱动，环保节能，亮度高，散射角度大，一致性好，超低功率及超长寿命等优点，将控制电路集成于LED上面，电路变得更加简单，体积小，安装更加简便。

机械尺寸（单位mm）



SOLOMON SYSTECH
SEMICONDUCTOR TECHNICAL DATA

SSD1306

Advance Information

128 x 64 Dot Matrix
OLED/PLED Segment/Common Driver with Controller

泽耀科技

A39C-T400A22D1a 产品手册

410-525MHz，158mW，LORA 扩频无线串口模块，
AES 加密，定点传输

版本：Spec_A39C-T400A22D1a _V9.3
日期：2022-9-30
状态：受控状态

泽耀科技

A39C 系列

快速操作指南

Easy to use guide manual for beginners

邮箱：support@ashining.com
官网：www.ashining.com
地址：四川省 成都市 高新西区百草路 808 号
智能信息产业园 2 层、5 层

ILITEK
Enrich visual experience

ILI9486

a-Si TFT LCD Single Chip Driver
320RGBx480 Resolution and 262K-color

Specification
Preliminary

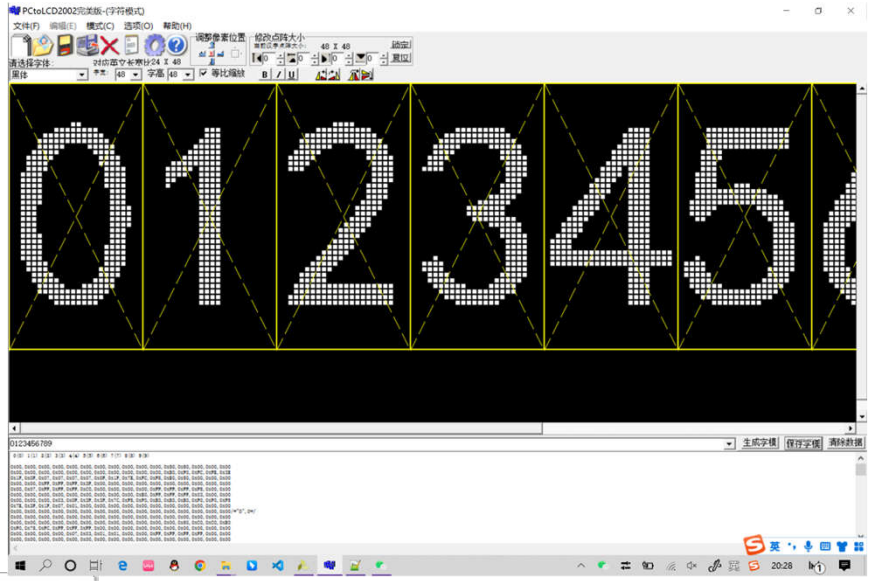
Version: V0.01
Document No: ILI9486_SPEC_V001.pdf

ILI TECHNOLOGY CORP.
8F, No. 38, Taiyuan St, Jhubei City,
Taiwan 302, R.O.C.
Tel.886-3-5600099; Fax.886-3-5600585
http://www.ilitek.com

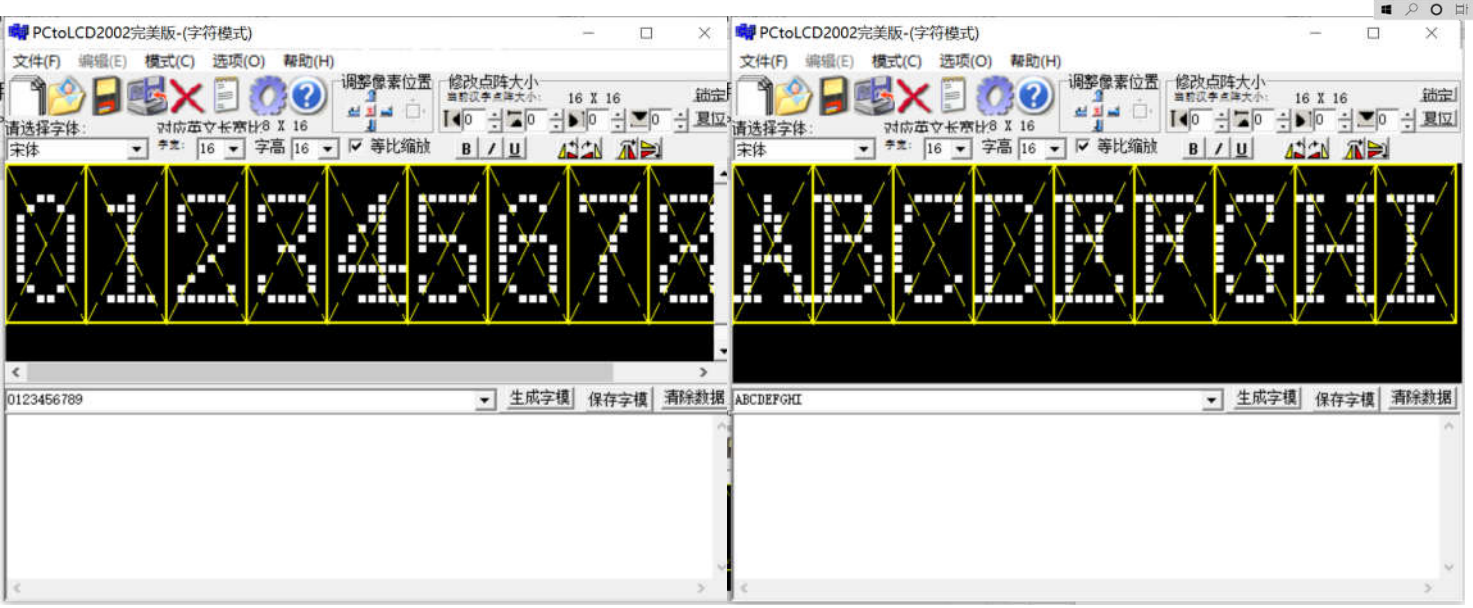
字体取模

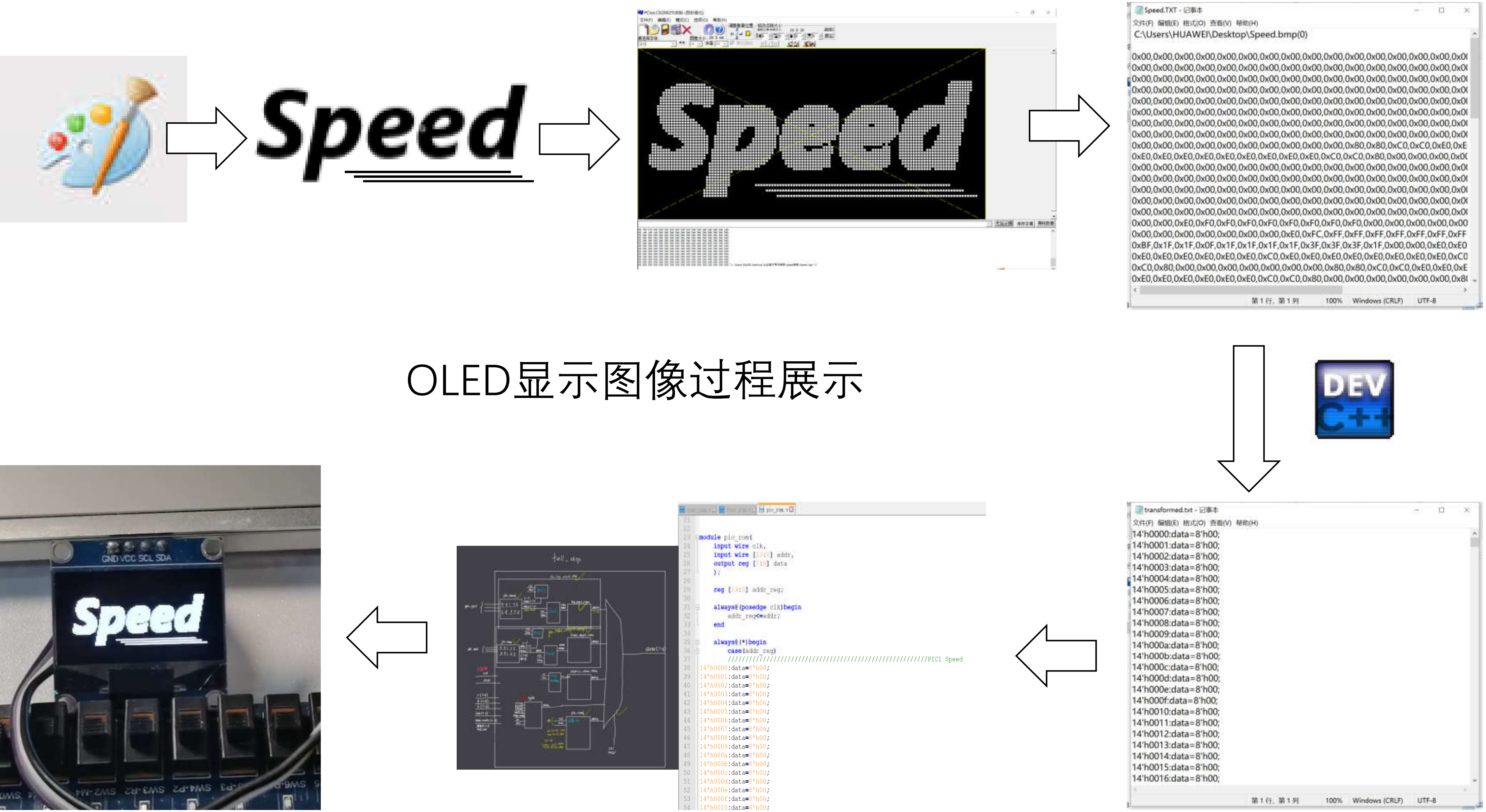
使用工具：PCtoLCD2002

24*48数字取模(for OLED)



8*16字母数字取模(for LCD OLED)





OLED显示图像过程展示

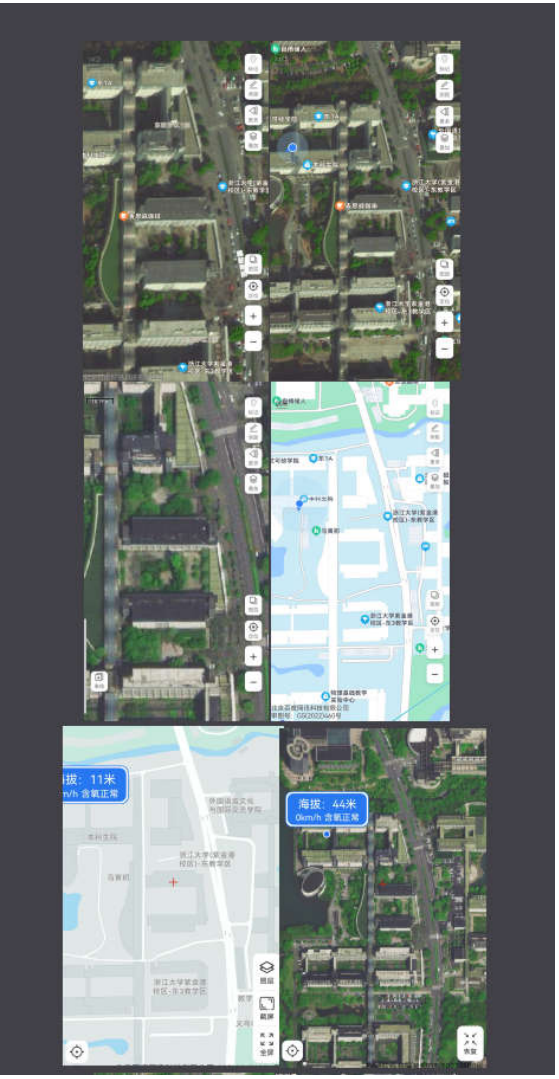
借助PCD2LCD2002和C
极大地提高了取模效率

格式转换部分程序

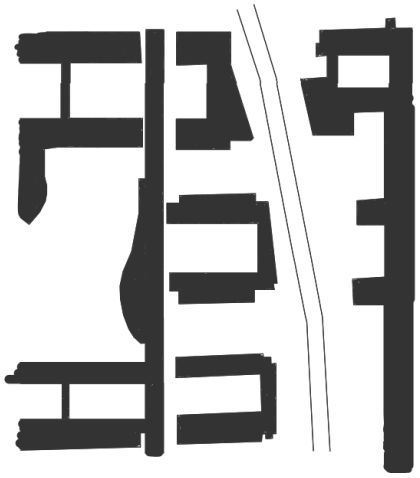
```
25  row=0;col=1024;
26  while(!feof(fp1))
27  {
28      ch=fgetc(fp1);
29
30      if(ch!=EOF){
31          if(ch=='x'){
32              a=fgetc(fp1);
33              b=fgetc(fp1);
34              addr=row*2048+col;
35              fprintf(fp2,"14'h%04x:data=8'h%c%c;\n",addr,a,b);
36              if(col==1151){
37                  col=1024;
38                  if(row==7){
39                      row=0;
40                      fprintf(fp2,"\n");
41                  }else{
42                      row++;
43                  }
44              }else{
45                  col++;
46              }
47          }
48      }
49  }
```

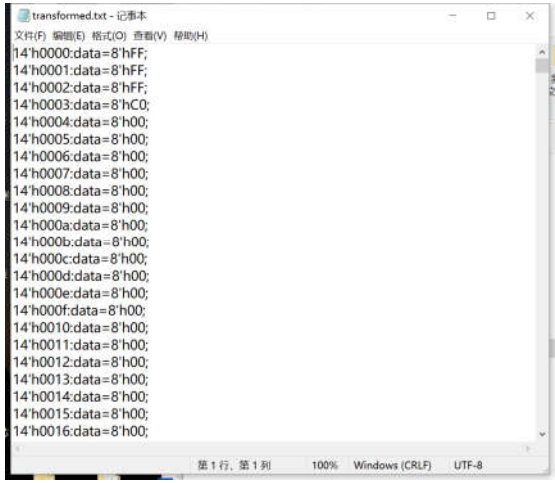
取模部分配置

东教区域地图获取过程


海拔：11米
海拔：44米


浙江大学(紫金港校区)-东教学区
浙江大学紫金港校区-东3教学楼
全屏




transformed.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
14'h0000:data=8'hFF;
14'h0001:data=8'hFF;
14'h0002:data=8'hFF;
14'h0003:data=8'hC0;
14'h0004:data=8'h00;
14'h0005:data=8'h00;
14'h0006:data=8'h00;
14'h0007:data=8'h00;
14'h0008:data=8'h00;
14'h0009:data=8'h00;
14'h000a:data=8'h00;
14'h000b:data=8'h00;
14'h000c:data=8'h00;
14'h000d:data=8'h00;
14'h000e:data=8'h00;
14'h000f:data=8'h00;
14'h0010:data=8'h00;
14'h0011:data=8'h00;
14'h0012:data=8'h00;
14'h0013:data=8'h00;
14'h0014:data=8'h00;
14'h0015:data=8'h00;
14'h0016:data=8'h00;
第 1 行，第 1 列
100%
Windows (CRLF)
UTF-8

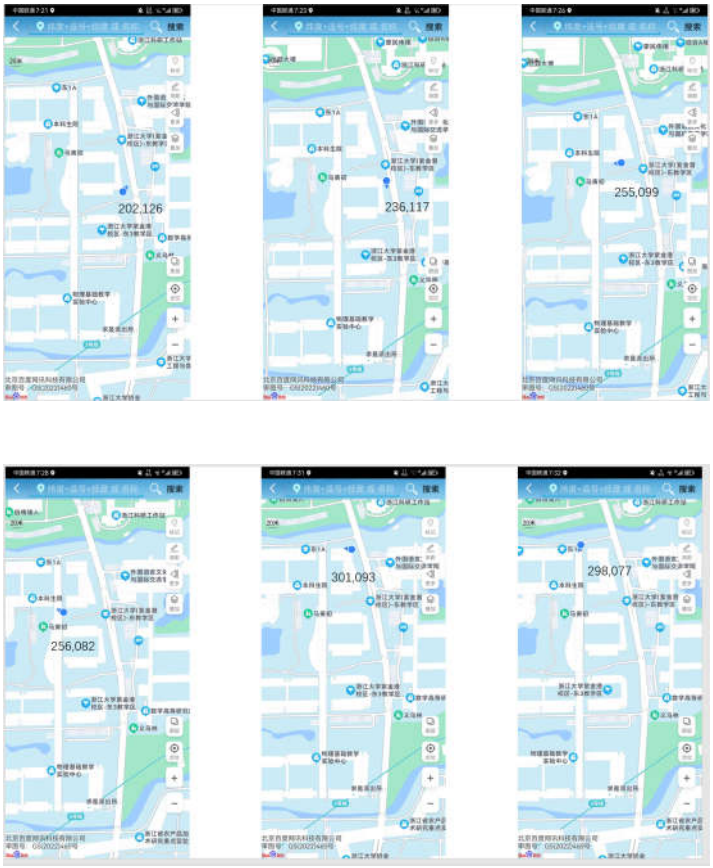
GPS定位信息采集及位置校准过程

实地坐标采集
每个点采样三次并求均值

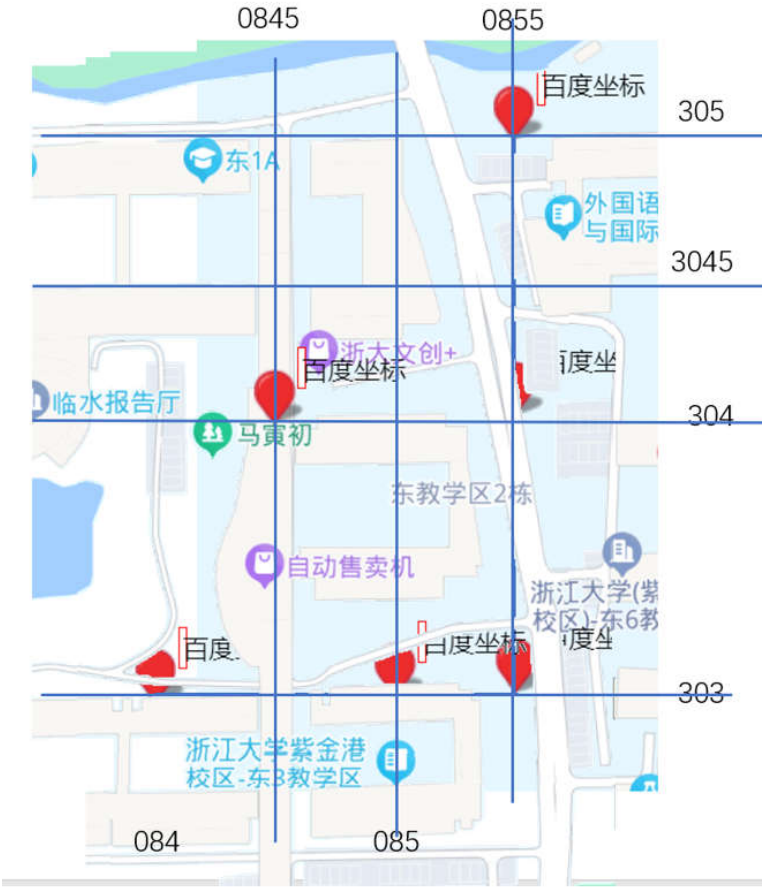
20230605GPS定位收集.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
1
$GNGGA,111418.000,3018.18672,N,12005.12940,E,1,18,0.7,15.8,M,0.0,M,,*44
$GNGGA,111450.000,3018.18754,N,12005.12930,E,1,20,0.7,17.5,M,0.0,M,,*4E
$GNGGA,111521.000,3018.18861,N,12005.12981,E,1,21,0.7,14.9,M,0.0,M,,*44

18762  12950

2
$GNGGA,111712.000,3018.19458,N,12005.10601,E,1,17,0.8,11.7,M,0.0,M,,*45
$GNGGA,111729.000,3018.19409,N,12005.10333,E,1,16,0.8,15.1,M,0.0,M,,*4E
$GNGGA,111751.000,3018.19473,N,12005.10230,E,1,17,0.8,14.7,M,0.0,M,,*48

19447  10388
3
$GNGGA,111856.000,3018.20102,N,12005.08745,E,1,15,0.8,17.0,M,0.0,M,,*41
$GNGGA,111914.000,3018.20013,N,12005.08589,E,1,16,0.8,21.9,M,0.0,M,,*4A
$GNGGA,111939.000,3018.19917,N,12005.08382,E,1,13,1.1,21.5,M,0.0,M,,*4E

20010  08572
4
$GNGGA,112139.000,3018.20344,N,12005.12612,E,1,23,0.6,26.3,M,0.0,M,,*40
$GNGGA,112157.000,3018.20196,N,12005.12619,E,1,23,0.8,27.9,M,0.0,M,,*4B
$GNGGA,112215.000,3018.20045,N,12005.12692,E,1,23,0.8,29.2,M,0.0,M,,*47

20195  12641
5
$GNGGA,112404.000,3018.23565,N,12005.11862,E,1,22,0.7,9.2,M,0.0,M,,*7B
$GNGGA,112426.000,3018.23601,N,12005.11682,E,1,21,0.7,8.0,M,0.0,M,,*7A
$GNGGA,112441.000,3018.23615,N,12005.11668,E,1,23,0.7,8.2,M,0.0,M,,*7A

23594  11737
6
$GNGGA,112619.000,3018.25578,N,12005.09867,E,1,19,0.7,8.7,M,0.0,M,,*7F
$GNGGA,112640.000,3018.25485,N,12005.09961,E,1,17,0.8,11.9,M,0.0,M,,*40
$GNGGA,112655.000,3018.25474,N,12005.09928,E,1,19,0.7,11.7,M,0.0,M,,*48

25512  09919
7
$GNGGA,112838.000,3018.25525,N,12005.08761,E,1,19,0.8,3.1,M,0.0,M,,*70
$GNGGA,112855.000,3018.25604,N,12005.08122,E,1,18,0.9,-2.4,M,0.0,M,,*53
$GNGGA,112912.000,3018.25633,N,12005.07636,E,1,18,0.8,-4.4,M,0.0,M,,*5F
```

标注各采样点位置及数据



图层叠加并获得映射比例及整体偏移量



重要模块介绍：纯组合电路专用乘法器

实际问题：对于串口传过来的数据，要想换算成可以用于计算的坐标，需要格式转换

串口数据示例：1 2 3 4 5 (char)

0x31 0x32 0x33 0x34 0x35 (ASCII)

-0x30

0x01 0x02 0x03 0x04 0x05

这是5字节的十进制数据，而且是串行接收的，需要转换成一个并行十进制数据才能用于加减和计算

最直接的方法：
0x01*10000+0x02*1000+0x03*100+0x04*10+0x05

但是乘法运算符是不可综合的，综合的话也可能会综合出一个乘法器，整个系统的时序也变得不受我们控制

所以我们需要一个纯组合电路，能够立即完成上面的计算

对于FPGA来说，最稳妥的办法是基于移位运算，也就是只能乘2或除2，但是乘10乘100并不能直接移位

最终我们采用移位和加减达到了这个目的：

1 *10000 +2 *1000 +3 *100 +4 *10 +5

1*(8192+2048-256+16)+2*(1024-16-8)+3*(64+36+4)+4*(8+2)+5

(1<<13)+(1<<11)-(1<<8)+(1<<4)+(2<<10)-(2<<4)-(2<<3)+(3<<6)+(3<<5)+(3<<2)+(4<<3)+(4<<1)+5

```
always@(*)
begin
data17=a[34:28]-7'h30;
data1=a[27:21]-7'h30;
data2=a[20:14]-7'h30;
data3=a[13:7]-7'h30;
data4=a[6:0]-7'h30;

data18={data17,13'b0000000000000};
data19={data17,11'b00000000000};
data20={data17,8'b00000000};
data21={data17,4'b0000};
data5={data1,10'b0000000000};
data6={data1,4'b0000};
data7={data1,3'b000};
data8={data2,6'b000000};
data9={data2,5'b00000};
data10={data2,2'b00};
data11={data3,3'b000};
data12={data3,1'b0};
data13=data5-data6-data7;
data14=data8+data9+data10;
data15=data11+data12;
data16=data18+data19-data20+data21;
data22=data13+data14+data15+data16+data4;
end

assign  out=data22[16:0];
```

再加之一个循环移位寄存器就可以满足系统的要求了

```
always@(posedge clk)begin
    if(rst)begin
        data_en_reg<=1'b0;
        state_reg<=st_0;
        jing1<=0;jing2<=0;jing3<=0;jing4<=0;jing5<=0;
        wei1<=0;wei2<=0;wei3<=0;wei4<=0;wei5<=0;
    end
    else begin
        jing1<=jing1_next;
        jing2<=jing2_next;
        jing3<=jing3_next;
        jing4<=jing4_next;
        jing5<=jing5_next;

        wei1<=wei1_next;
        wei2<=wei2_next;
        wei3<=wei3_next;
        wei4<=wei4_next;
        wei5<=wei5_next;

        state_reg<=state_next;
        data_en_reg<=data_en_next;
    end
end
```

```
st_0:begin
    if(gps_we)begin
        wei1_next=gps_data;
        state_next=st_1;
    end
end
st_1:begin
    if(gps_we)begin
        wei2_next=gps_data;
        state_next=st_2;
    end
end
st_2:begin
    if(gps_we)begin
        wei3_next=gps_data;
        state_next=st_3;
    end
end
st_3:begin
    if(gps_we)begin
        wei4_next=gps_data;
        state_next=st_4;
    end
end
st_4:begin
    if(gps_we)begin
        wei5_next=gps_data;
        state_next=st_6;
    end
end
```

重要模块介绍：GPS信息提取解读模块（通用）

GPS模块原始数据（NEMA-0183协议）

定位成功：

$GNGGA,133202.000,3018.59179,N,12004.58755,E,6,04,2.6,-12.0,M,0.0,M,,*6E↵
$GNGLL,3018.59179,N,12004.58755,E,133202.000,A,E*46↵
$GPGSA,A,3,13,15,18,24,,,,,,,,,6.3,2.6,5.8*37↵
$BDGSA,A,3,,,,,,,,,,,,,6.3,2.6,5.8*2F↵
$GPGSV,2,1,05,13,29,043,22,15,56,023,24,18,59,293,28,20,13,113,*75↵
$GPGSV,2,2,05,24,63,167,21*4C↵
$BDGSV,1,1,00*68↵
$GNRMC,133202.000,A,3018.59179,N,12004.58755,E,2.41,108.45,120523,,,E*79↵
$GNVTG,108.45,T,,M,2.41,N,4.47,K,E*2F↵
$GNZDA,133202.000,12,05,2023,00,00*4C↵
$GPTXT,01,01,01,ANTENNA OK*35↵

定位失败：

$GNGGA,133200.000,,,,,0,00,25.5,,,,,,*79↵
$GNGLL,,,,,133200.000,V,M*64↵
$GPGSA,A,1,,,,,,,,,,,,,25.5,25.5,25.5*02↵
$BDGSA,A,1,,,,,,,,,,,,,25.5,25.5,25.5*13↵
$GPGSV,2,1,05,13,29,043,22,15,56,023,24,18,59,293,20,20,13,113,*7D↵
$GPGSV,2,2,05,24,63,167,21*4C↵
$BDGSV,1,1,00*68↵
$GNRMC,133200.000,V,,,,,,120523,,,M*54↵
$GNVTG,,,,,,,,,M*2D↵
$GNZDA,133200.000,12,05,2023,00,00*4E↵
$GPTXT,01,01,01,ANTENNA OK*35↵

我们需要提取和利用的信息：

定位成功：↵
$GNRMC,133245.000,A,3018.58806,N,12004.59639,E,1.42,60.85,120523,,,A*47↵
$GNVTG,60.85,T,,M,1.42,N,2.62,K,A*19↵
定位失败：↵
$GNRMC,133200.000,V,,,,,,120523,,,M*54↵
$GNVTG,,,,,,,,,M*2D↵

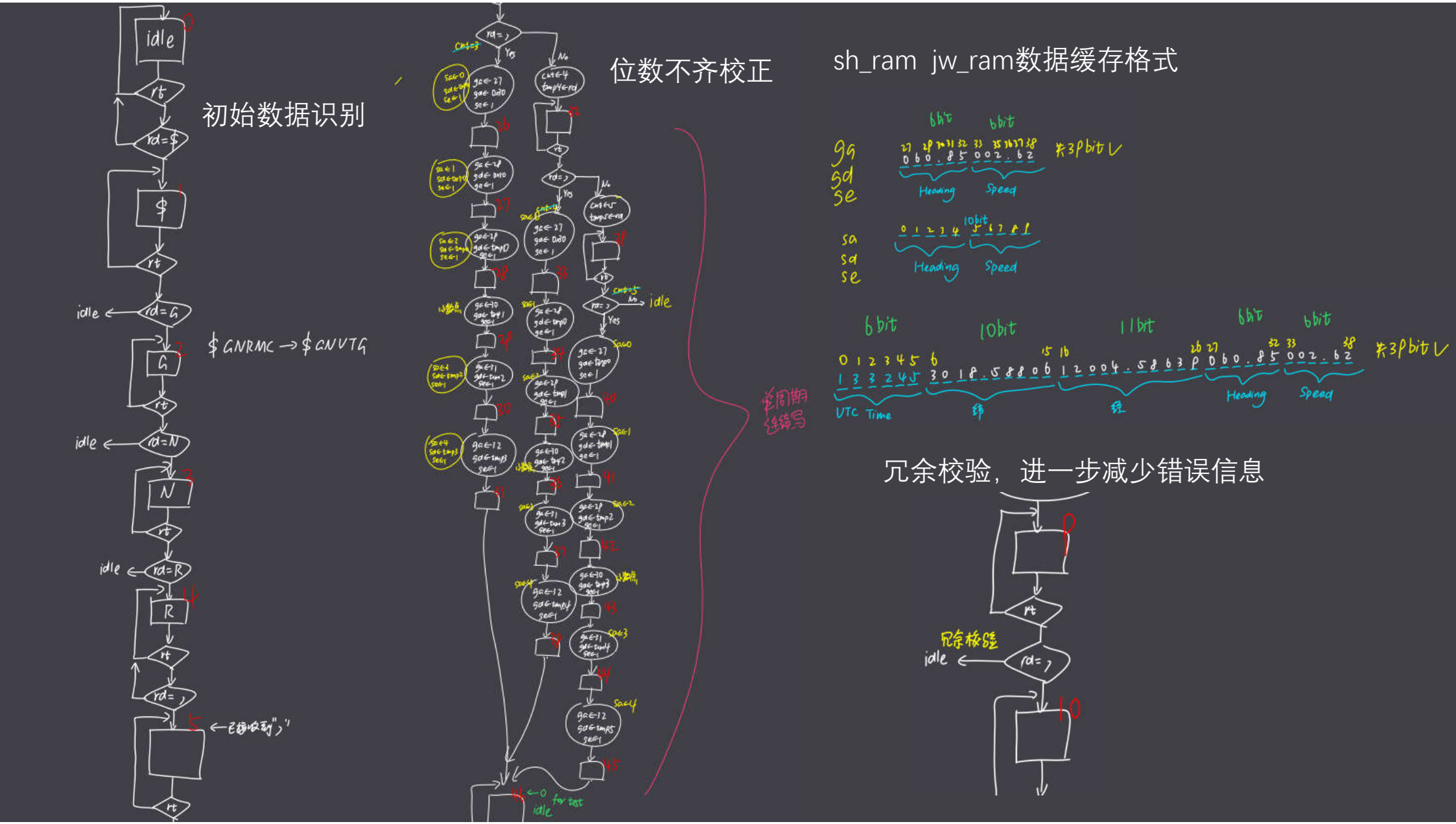
难点：

1.需要在这一坨的数据里面准确找到想要的信息并锁存下来

2.每次发送数据的位数和格式都是不一样的

3.航向和航速的位数不确定（前面不会补0）无法直接利用

4.由于实际无线传输的时候串口信息可能会不稳定，需要错误发现并丢弃数据的机制，尽量减少输入到系统中错误数据的个数


初始数据识别
$GNRMC → $GNVTG
位数不齐校正
sh_ram jw_ram数据缓存格式
6bit
6bit
Heading
Speed
10bit
Heading
Speed
6bit
10bit
11bit
6bit
6bit
UTC Time
Heading
Speed
冗余校验，进一步减少错误信息
idle

关于RAM与ROM

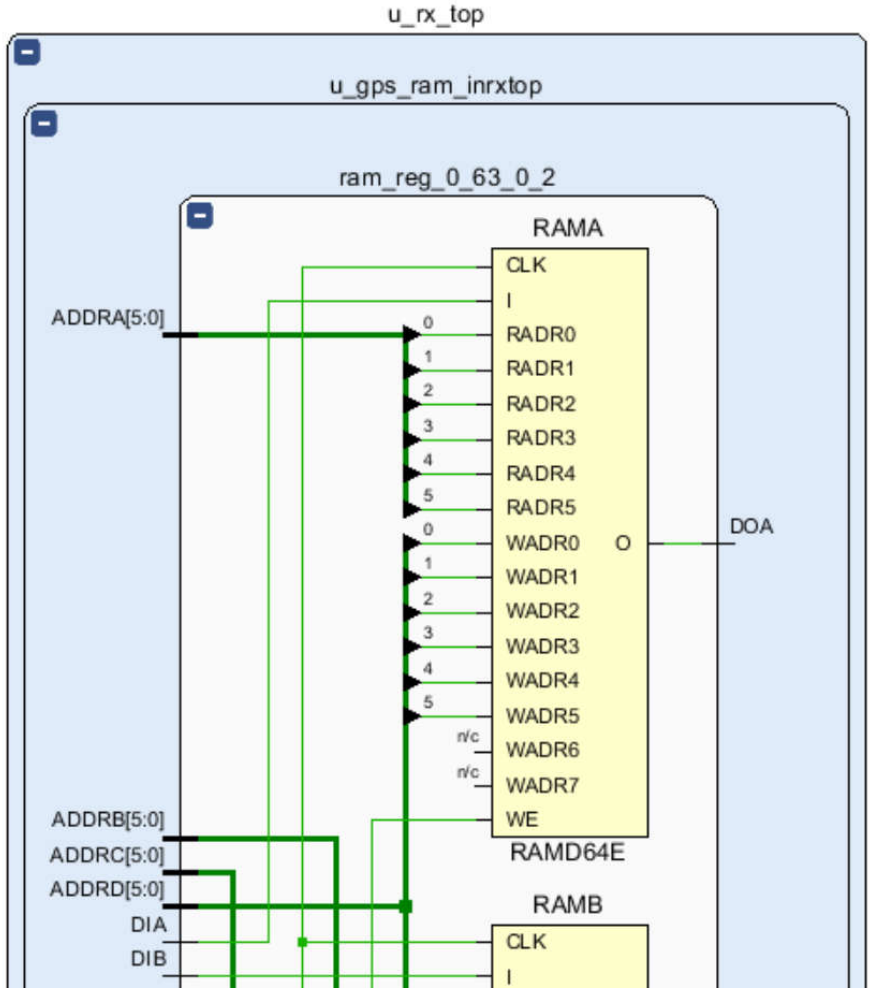
本项目由于各模块在时间上不同步，且需要信息传递，所以需要各种数据的缓存，用到大量的RAM

通过采用较为标准的写法，可以被稳定地综合出块RAM和基于块RAM的ROM

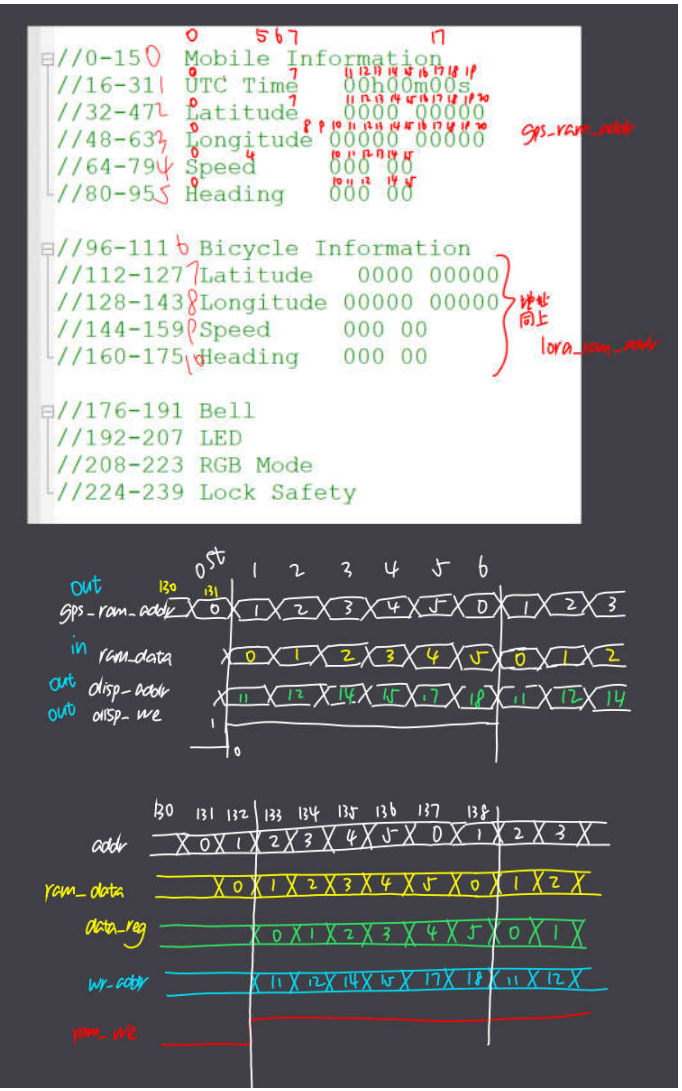
项目初期担心板子资源不足，通过这种方式可以减少逻辑资源的开销，后来发现担心是多余的，但是这种写法确实方便

通用双端口（端口A读写，端口B只读）RAM

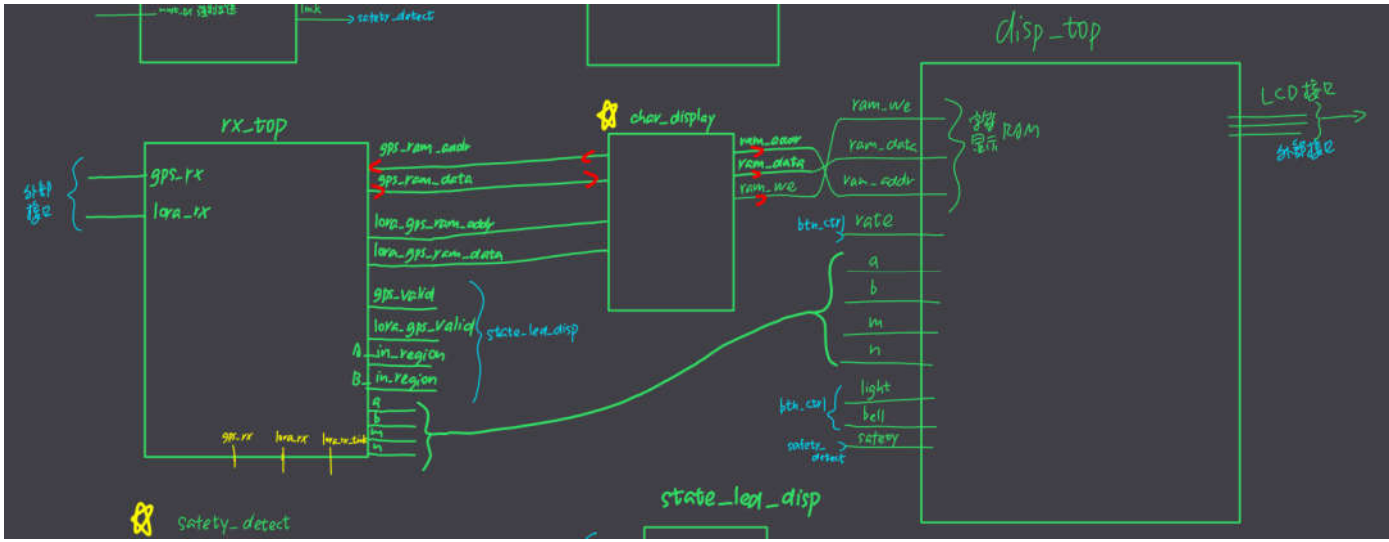
```
module gps_ram
#(
    parameter ADDR_WIDTH=6,//39MAX  [5:0] addr
    DATA_WIDTH=8
)
(
    input wire clk,
    input wire we,
    input wire [ADDR_WIDTH-1:0] addr_a,addr_b,//a,b地址端口
    input wire [DATA_WIDTH-1:0] din_a,//a端口写入数据
    output wire [DATA_WIDTH-1:0] dout_a,dout_b
    );

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

    reg [ADDR_WIDTH-1:0] addr_a_reg,addr_b_reg;//端口地址缓存

    //写功能
    always@(posedge clk)begin
        if(we)begin
            ram[addr_a]<=din_a;
        end
        addr_a_reg<=addr_a;
        addr_b_reg<=addr_b;
    end

    //读功能
    assign dout_a=ram[addr_a_reg];
    assign dout_b=ram[addr_b_reg];

endmodule
```



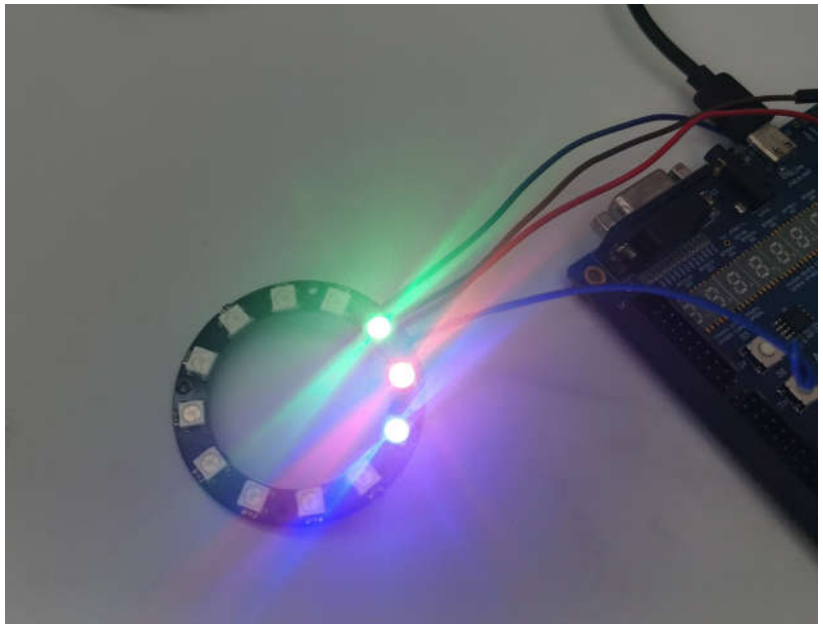
重要模块介绍：RAM数据快速整合转移模块及固定字符位置映射模块




```
//数据流准备阶段
st_130:begin state_next=st_131;gps_ram_addr_next=0;end
st_131:begin state_next=st_132;gps_ram_addr_next=1;end
st_132:begin state_next=st_133;gps_ram_addr_next=2;ram_we_next=1'b1;
//////////////////////////////////////////////////////////////////////////////UTC Time
    row_next=1;col_next=11;//0
    data_next=gps_ram_data;end
//数据流循环开始
st_133:begin state_next=st_134;gps_ram_addr_next=3;ram_we_next=1'b1;
    row_next=1;col_next=12;//1
    data_next=gps_ram_data;end
st_134:begin state_next=st_135;gps_ram_addr_next=4;ram_we_next=1'b1;
    row_next=1;col_next=14;//2
    data_next=gps_ram_data;end
st_135:begin state_next=st_136;gps_ram_addr_next=5;ram_we_next=1'b1;
    row_next=1;col_next=15;//3
    data_next=gps_ram_data;end
st_136:begin state_next=st_137;gps_ram_addr_next=6;ram_we_next=1'b1;
    row_next=1;col_next=17;//4
    data_next=gps_ram_data;end
st_137:begin state_next=st_138;gps_ram_addr_next=7;ram_we_next=1'b1;
    row_next=1;col_next=18;//5
    data_next=gps_ram_data;end
    //////////////////////////////////////////////////////////////////////////////Latitude
st_138:begin state_next=st_139;gps_ram_addr_next=8;ram_we_next=1'b1;
    row_next=2;col_next=11;//6
    data_next=gps_ram_data;end
st_139:begin state_next=st_140;gps_ram_addr_next=9;ram_we_next=1'b1;
    row_next=2;col_next=12;//7
    data_next=gps_ram_data;end
st_140:begin state_next=st_141;gps_ram_addr_next=10;ram_we_next=1'b1;
    row_next=2;col_next=13;//8
    data_next=gps_ram_data;end
```



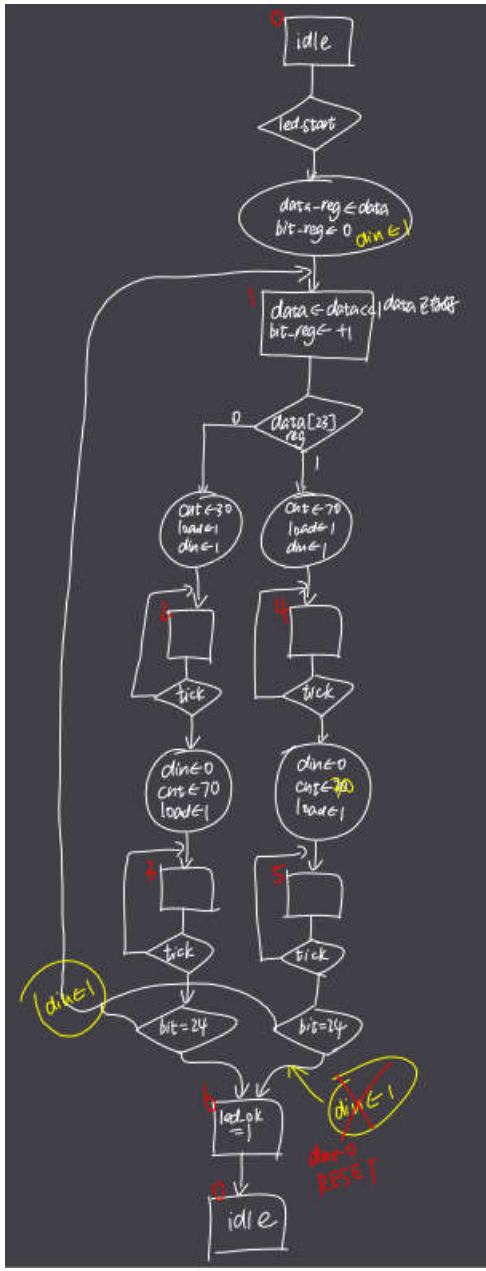
WS2812通用驱动开发（最终未应用到项目中）



```
module led_driver(
    input wire clk,rst,
    input wire [23:0] led_data,
    input wire led_start,
    input wire tick,
    output wire din,
    output reg led_ok,
    output wire [7:0] cnt,
    output wire load
    );
```

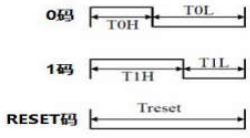
相较于开源的驱动，状态数更少，模块通用性更强

```
case(state_reg)
    st_idle:begin
        if(led_start)begin
            data_next=led_data;
            bit_next=0;
            state_next=st_1;
            din_next=1'b1;
        end
    end
    st_1:begin
        data_next=data_reg<<1;
        bit_next=bit_reg+1;
        if(data_reg[23])begin
            cnt_next=70;
            load_next=1'b1;
            state_next=st_4;
        end
        else begin
            cnt_next=30;
            load_next=1'b1;
            state_next=st_2;
        end
    end
    st_2:begin
        if(tick)begin
            state_next=st_3;
            din_next=1'b0;
            cnt_next=70;
            load_next=1'b1;
        end
    end
    st_3:begin
        if(tick)begin
            if(bit_reg==24)begin
                state_next=st_6;
                din_next=1'b0;
            end
            else begin
                state_next=st_1;
                din_next=1'b1;
            end

        end
    end
    st_4:begin
        if(tick)begin
            state_next=st_5;
            din_next=1'b0;
            cnt_next=30;
            load_next=1'b1;
        end
    end
    st_5:begin
        if(tick)begin
            if(bit_reg==24)begin
                state_next=st_6;
                din_next=1'b0;
            end
            else begin
                state_next=st_1;
```



时序波形图

输入码型：



24bit 数据结构

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

注：高位先发，按照 GRB 的顺序发送数据。

SSD1306专用显示IIC驱动开发

由于开源的IIC驱动是为了众多器件设计的，所以对于OLED的驱动存在功能冗余，我们也没有找到很好的适用于SSD1306的驱动，由于IIC本身的SCL频率存在限制(标准400KHz), 测试之后发现滚动动画十分卡顿，于是我们开发了专用于OLED显示的IIC驱动，去掉了没有必要的传输环节，并且通过超频(800KHz)，最终实现了较为流畅的滚动显示。