一、项目需求：

该项目基于小脚丫FPGA的综合技能训练板，利用ADC制作一个数字电压表，具体功能如下：

1、旋转电位计可以产生0-3.3V的电压；

2、利用板上的串行ADC对电压进行转换；

3、将电压值在板上的OLED屏幕上显示出来；

4、通过按键可以锁定电压值，相当于万用表上面端的HOLD功能；

二、设计思路：

小脚丫FPGA的综合技能训练板上带有一颗SPI接口的串行ADC，可以采集电位计上的电压。板上的ADC是串行ADC，采样率最高为280ksps，可以对频率为20KHz以内的信号（音频信号的范围）进行采样；同时，板上采用了一块128\*32分辨率的OLED作为信息显示终端，可以将ADC采集到的电压值显示在屏幕上。板上的OLED和ADC器件都使用SPI通信，因此该系统实现的核心在于解决SPI通信。

总体设计模块如下图1所示：

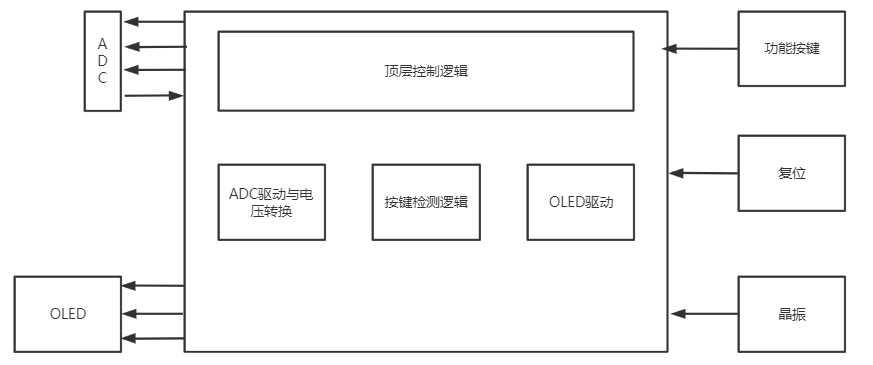


图1 总体设计模块图

三、模块实现：

本系统由ADC驱动模块、OLED驱动模块、ADC电压计算模块、功能按键实现模块和顶层总体逻辑控制模块，分别如下所述：

1、ADC驱动模块：

小脚丫FPGA的综合技能训练板上所带的ADC是一颗BB公司生产的8bit-串行ADC，该ADC最大可以达到280KSPS采样率，该ADC框图如图2所示：

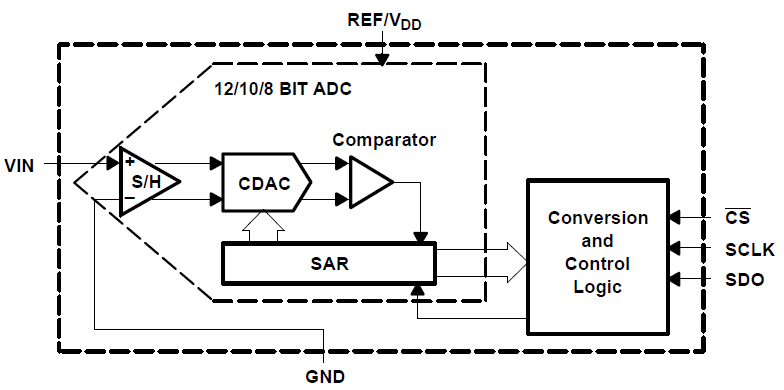


图2 ADS7868框图

根据ADS7868手册提供的时序图可知该ADC的驱动SPI基本符合标准SPI通信，如图3所示：

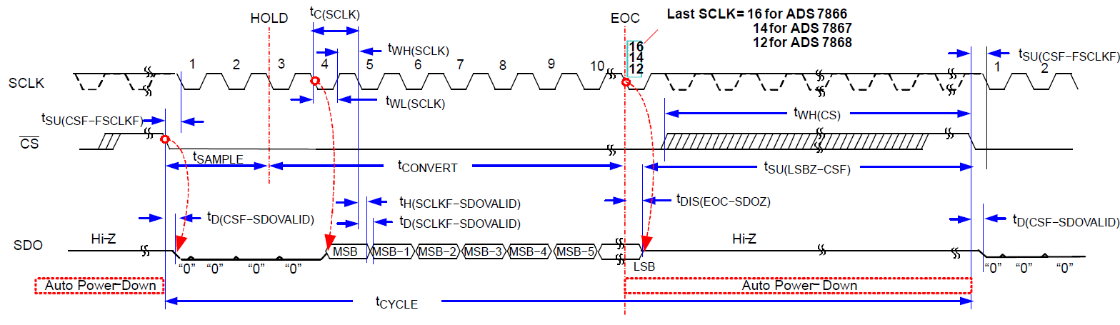


图3 ADS7868时序图

根据该时序图，设计ADC的驱动状态机如图4所示：

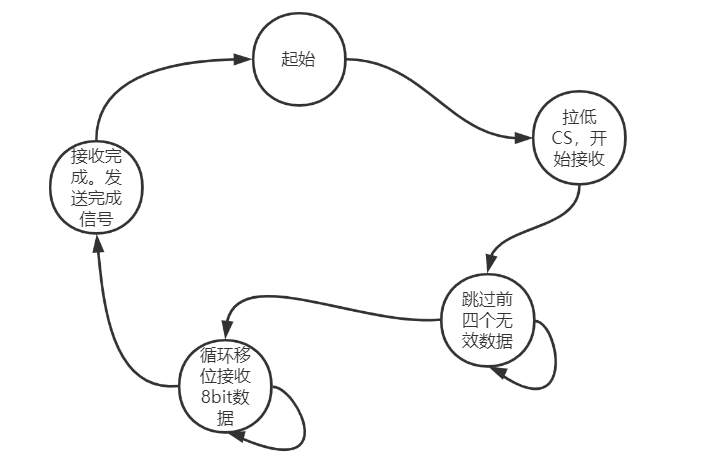


图4 ADC驱动状态机

根据该状态机的状态转移设置，且由于输入时钟较快，因此采用分频代码对输入时钟进行8分频，使得ADS7868的驱动时钟符合datasheet中的时钟要求（<6.7us，即最大频率不超过149KHz）编写代码如下所示：

1. /\*分频产生fclk\*/
2. always @(posedge I\_sys\_clk or negedge I\_rst\_n) begin
3. **if**(~I\_rst\_n) begin
4. f\_sck <= 0;
5. end
6. **else** begin
7. **if**(R\_cnt == PERIOD-1) begin
8. f\_sck <= ~f\_sck;
9. R\_cnt <= 0;
10. end
11. **else** begin
12. R\_cnt <= R\_cnt + 1;
13. end
14. end
15. end
17. /\*采样状态机\*/
19. always @(posedge I\_sys\_clk or negedge I\_rst\_n)
20. **if**(!I\_rst\_n) begin
21. O\_adc\_cs <= 1'b1; O\_adc\_clk <= 1'b1;
22. O\_data <=0;
23. O\_done <=1'b0;
24. bit\_cnt <= 4'd8;
25. nop\_cnt <= 4'd0;
26. R\_state <= 4'd0;
27. end
28. **else** **case**(R\_state)
29. 8'd0 :  begin
30. O\_adc\_cs <= 1'b1;
31. O\_adc\_clk <= 1'b1;
32. R\_state <= 4'd1;
33. bit\_cnt <= 4'd8;
34. nop\_cnt <= 4'd0;
35. end
36. 8'd1 :  begin
37. O\_done <= 1'b0;
38. **if**(nop\_cnt == 4'd4) begin
39. R\_state <= 4'd3;
40. end
41. **else** begin
42. O\_adc\_cs <= 1'b0;
43. O\_adc\_clk <= 1'b1;
44. R\_state <= 4'd2;
45. end
46. end
47. 8'd2 :  begin
48. O\_adc\_cs <= 1'b0;
49. O\_adc\_clk <= 1'b0;
50. nop\_cnt <= nop\_cnt + 4'd1;
51. R\_state <= 4'd1;
52. end
53. 8'd3 :  begin
54. **if**(bit\_cnt == 0) begin
55. R\_state <= 4'd5;
56. end
57. **else** begin
58. O\_adc\_cs <= 1'b0;
59. O\_adc\_clk <= 1'b1;
60. O\_data[bit\_cnt-1] <= I\_adc\_data;
61. R\_state <= 4'd4;
62. end

65. end //4
66. 8'd4 :  begin
67. O\_adc\_cs <= 1'b0;
68. O\_adc\_clk <= 1'b0;
69. bit\_cnt <= bit\_cnt - 4'd1;
70. R\_state <= 4'd3;
71. end
72. 8'd5 :  begin
73. O\_adc\_cs <= 1'b1;
74. O\_adc\_clk <= 1'b0;
75. O\_done <= 1'b1;
76. R\_state <= 4'd0;
77. end
78. **default** : begin
79. O\_adc\_cs <= 1'b1;
80. O\_adc\_clk <= 1'b1;
81. O\_done <= 1'b0;
82. end
83. endcase

2、OLED驱动模块：

OLED使用FPGA驱动较为复杂，该128X32的点阵屏幕采用SSD1306，虽然SSD1306有多种驱动方式，但是在板子上该驱动方式仅为SPI驱动实现，4-wire SPI总线驱动时序图如图5所示，图6显示为SDD1306刷新RAM时地址与数据的分配关系：

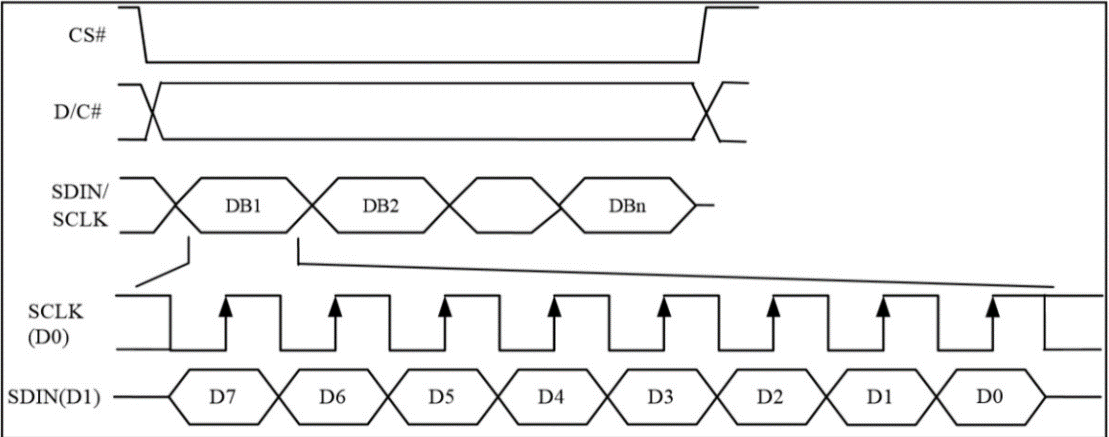


图5 4-wire SPI驱动时序图

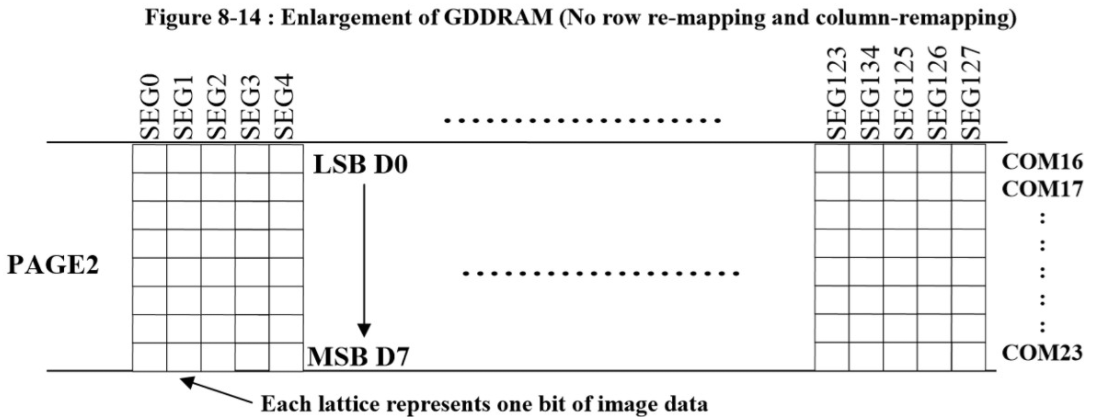


图6 地址与数据位分配关系

由于能力有限，且为了加快设计进度，在电子森林上找到了相关的驱动程序（项目地址见【https://www.eetree.cn/wiki/oled\_spi\_verilog】），但是该驱动程序是直接驱动方式，数据和位置都写死在程序中，可移植性和灵活性较差，因此，对该驱动程序做适当修改，将其改为RAM读写的驱动端口，这样，就可以在RAM的另一个写端口将要显示的内容按地址写入，随后，OLED驱动程序在RAM的读端口按地址读取要显示的数据。修改接口程序如下，提供ram的读地址和读信号：

1. **case**(cnt\_main)  //MAIN状态
2. 5'd0:   begin state <= INIT; end
3. 5'd1:begin
4. ram\_addr <= 4'd1;
5. end
6. 5'd2:   begin  y\_p <= 8'hb0; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; **char** <= ram\_data[16\*8-1:0];state <= SCAN; end
7. 5'd3: begin
8. ram\_addr <= 4'd2;
9. end
10. 5'd4:   begin  y\_p <= 8'hb1; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; **char** <= ram\_data[16\*8-1:0];state <= SCAN; end
11. 5'd5:begin
12. ram\_addr <= 4'd3;
13. end
14. 5'd6:   begin  y\_p <= 8'hb2; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; **char** <= ram\_data[16\*8-1:0];state <= SCAN; end
15. 5'd7:begin
16. ram\_addr <= 4'd0;
17. end
18. 5'd8:   begin  y\_p <= 8'hb3; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; **char** <= ram\_data[16\*8-1:0];state <= SCAN; end
19. // 5'd1:    begin y\_p <= 8'hb0; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; char <= "OLED TEST       ";state <= SCAN; end
20. // 5'd2:    begin y\_p <= 8'hb1; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; char <= "OLED TEST       ";state <= SCAN; end
21. // 5'd3:    begin y\_p <= 8'hb2; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; char <= "OLED TEST       ";state <= SCAN; end
22. // 5'd4:    begin y\_p <= 8'hb3; x\_ph <= 8'h10; x\_pl <= 8'h00; num <= 5'd16; char <= "OLED TEST       ";state <= SCAN; end
24. // 5'd5:    begin y\_p <= 8'hb0; x\_ph <= 8'h15; x\_pl <= 8'h00; num <= 5'd 1; char <= sw; state <= SCAN; end
25. // 5'd6:    begin y\_p <= 8'hb1; x\_ph <= 8'h15; x\_pl <= 8'h00; num <= 5'd 1; char <= sw; state <= SCAN; end
26. // 5'd7:    begin y\_p <= 8'hb2; x\_ph <= 8'h15; x\_pl <= 8'h00; num <= 5'd 1; char <= sw; state <= SCAN; end
27. // 5'd8:    begin y\_p <= 8'hb3; x\_ph <= 8'h15; x\_pl <= 8'h00; num <= 5'd 1; char <= sw; state <= SCAN; end
29. **default**: state <= IDLE;
30. endcase

最后，修改后的的OLED程序部分如下所示，RAM的配置的log如图7所示：

1. GRAM\_MY inst0\_0 (
2. .WrAddress( WrAddress),
3. .RdAddress( RdAddress),
4. .Data(Data ),
5. .WE(1'b1 ),
6. .RdClock(I\_sys\_clk ),
7. .RdClockEn( 1'b1),
8. .Reset(!I\_rst\_n ),
9. .WrClock(I\_sys\_clk ),
10. .WrClockEn(1'b1 ),
11. .Q( ram\_data)
12. );

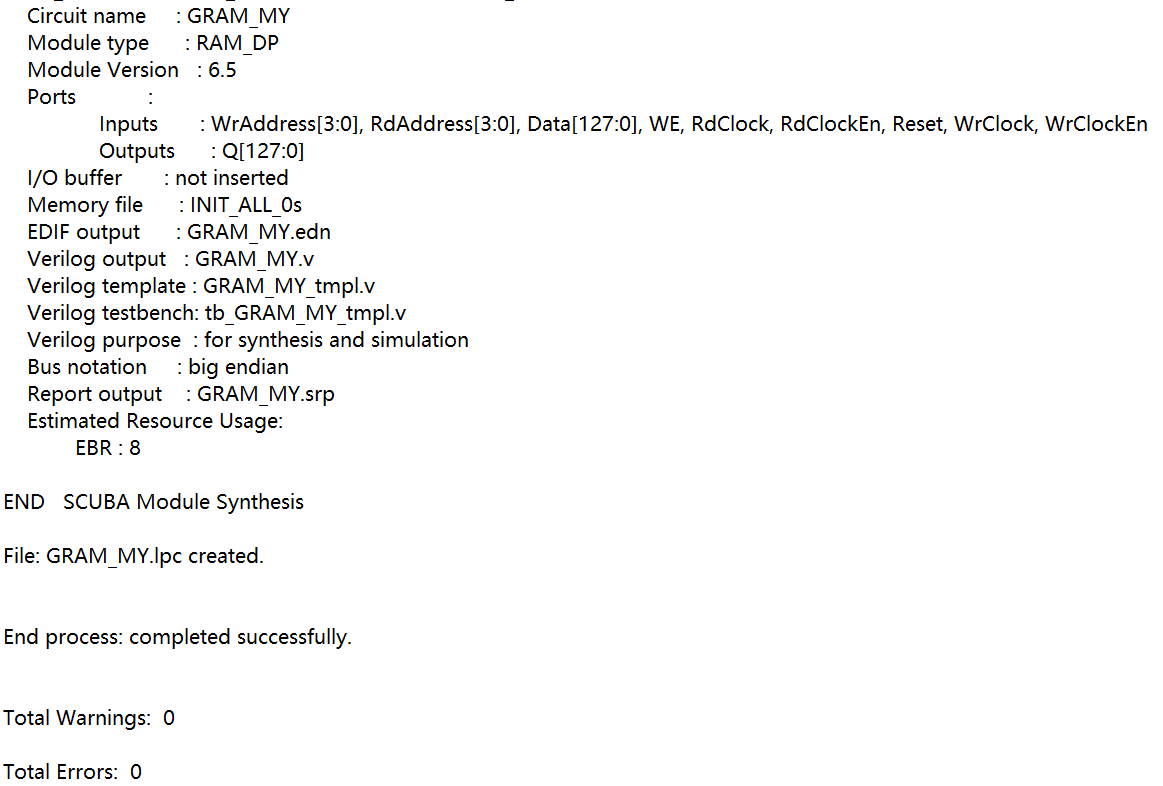


图7 OLED的RAM配置log

3、ADC电压计算模块：

由于ADC采集后的结果为8bit的二进制数，如果要显示在OLED上，需要将该二进制数使用如下公式1计算最后得到的电压值，然后采用移位法将获得电压值转换为BCD数，并分为小数和整数显示：

ADC\_Volt = 3.3X(CODE /255) (公式1)

设定ADC电压的显示为5个部分，第一部分为整数部分，第二部分为小数点，第三到第五部分为小数部分，代码如下所示：

1. ADS7868\_DRIVER inst0(
2. .I\_sys\_clk(f\_sck),
3. .I\_rst\_n(I\_rst\_n),
5. .I\_adc\_data(I\_data),
6. .O\_adc\_clk(O\_adc\_clk),
7. .O\_adc\_cs(O\_adc\_cs),
9. .O\_done(O\_led),
10. .O\_data(O\_data)
11. );
13. wire [15:0] bin\_code = O\_data \* 16'd129 + 16'd010;
14. reg [35:0]  shift\_reg;
15. //reg [19:0] bcd\_code;
16. always@(bin\_code or I\_rst\_n)begin
17. shift\_reg = {20'h0,bin\_code};
18. **if**(!I\_rst\_n) bcd\_code = 0;
19. **else** begin
20. repeat(16) begin //循环16次
21. //BCD码各位数据作满5加3操作，
22. **if** (shift\_reg[19:16] >= 5) shift\_reg[19:16] = shift\_reg[19:16] + 2'b11;
23. **if** (shift\_reg[23:20] >= 5) shift\_reg[23:20] = shift\_reg[23:20] + 2'b11;
24. **if** (shift\_reg[27:24] >= 5) shift\_reg[27:24] = shift\_reg[27:24] + 2'b11;
25. **if** (shift\_reg[31:28] >= 5) shift\_reg[31:28] = shift\_reg[31:28] + 2'b11;
26. **if** (shift\_reg[35:32] >= 5) shift\_reg[35:32] = shift\_reg[35:32] + 2'b11;
27. shift\_reg = shift\_reg << 1;
28. end
29. bcd\_code = shift\_reg[35:16];
30. end
31. end

34. /\*BCD数拆分为一位整数，三位小数，用于OLED显示\*/
35. assign new\_bcd[39:32]  = {4'b0,bcd\_code[19:16]};
36. assign new\_bcd[31:24]  = 8'd46;
37. assign new\_bcd[23:16]  = {4'b0,bcd\_code[15:12]};
38. assign new\_bcd[15:8]  = {4'b0,bcd\_code[11:8]};
39. assign new\_bcd[7:0]  = {4'b0,bcd\_code[7:4]};

4、功能按键实现模块：

为了便于后面实现电压测量过程中方便记录电压，因此增加了一个额外功能：电压显示锁，该功能就是在按一次按键后，OLED 会显示“LOCKED”字样，此时OLED 的电压会锁定在按键前的一刻的采样电压值。

为了保证按键都是每次按下有效，因此采用边沿检测的方法，每当检测到按键上升沿的同时，根据前一状态判断是锁定电压值还是解锁电压值，代码如下所示：

1. /\*选择输出\*/
2. assign new\_bcd\_t = (key\_scan == 0) ? new\_bcd\_t : new\_bcd;

5. /\*按键上升沿检测\*/
6. reg [1:0] key\_buf;
7. wire W\_key\_pos;
9. always@(posedge I\_sys\_clk or negedge I\_rst\_n)
10. begin
11. **if**(~I\_rst\_n) begin
12. key\_buf <= 0;
13. end
14. **else** begin
15. key\_buf <= {key\_buf[0],I\_key\_lock};
16. end
17. end
19. assign W\_key\_pos = (key\_buf == 2'b01) ? 1'b1 : 1'b0;


23. always @(posedge I\_sys\_clk or negedge I\_rst\_n) begin
24. **if**(~I\_rst\_n) begin
25. key\_scan <= 0;
26. end
27. **else** begin
28. **if**(W\_key\_pos) begin
29. key\_scan <= ~key\_scan;
30. end
31. **else** begin
32. key\_scan <= key\_scan;
33. end
34. end
35. end

5、顶层总体逻辑控制模块：

顶层控制逻辑负责控制整体顺序，并负责传递各个模块的数据，最主要的就是负责把ADC计算好的电压BCD码组装到显示RAM中，并且OLED从RAM中取出显示数据，并显示在OLED上面，代码如下所示：

1. reg [3:0] R\_mainstate;
3. always @(posedge I\_sys\_clk or negedge I\_rst\_n) begin
4. **if**(~I\_rst\_n) begin
5. WrAddress <= 4'd0;
6. Data <= {"    VOLTMETER   "};
7. //WE <= 1'b0;
8. R\_mainstate <= 4'd0;
9. end
10. **else** begin
11. **case** (R\_mainstate)
12. 4'd0 : begin
13. Data <= {"    VOLTMETER   "};
14. WrAddress <= 4'd0;
15. //WE <= 1'b1;
16. R\_mainstate <= 4'd1;
17. end
18. 4'd1 :begin
19. Data <= {"voltage : ",new\_bcd\_t,"V"};
20. WrAddress <= 4'd1;
21. R\_mainstate <= 4'd2;
22. end
23. 4'd2 : begin
24. Data <= {"liuyunfeng  edit"};
25. WrAddress <= 4'd2;
26. //WE <= 1'b1;
27. R\_mainstate <= 4'd3;
28. end
29. 4'd3 :begin
30. **if**(~key\_scan) begin
31. Data <= {"     LOCKED     "};
32. end
33. **else** begin
34. Data <= {"     RUNNING    "};
35. end
37. WrAddress <= 4'd3;
38. R\_mainstate <= 4'd0;
39. end
40. **default**: begin
41. WrAddress <= 4'd0;
42. Data <= 128'd0;
43. //WE <= 1'b0;
44. R\_mainstate <= 4'd0;
45. end
46. endcase
47. end
48. end

四、测试：

分为三组测试内容，分别为电压扫描测试、电压锁定测试和电压对比测试，具体测试详见b站视频。

五、遇到问题：

在开发中遇到电压校准存在问题，且误差较大，需要后续进行再次校正；同时，由于OLED 采用他人开源代码，导致并不是能非常方便的上手修改与使用，后续会重新修改方便扩展使用。

六、项目总结：

在本次项目开发中，首次接触到lattice的fpga和他的EDA工具，刚上手废了一段时间学习新的FPGA，并开发测试；同时在完成本项目中更加深刻的学习了SPI通信和OLED 的驱动方式。