2020 年“TI 杯”第八届浙江省

大学生电子设计竞赛

设计报告

竞赛题号：E

竞赛题目：放大器非线性失真研究装置

参赛学生：钟沁轩 顾程鑫 李伟

参赛学校：杭州电子科技大学

2020 年 10 月

摘要

本作品完成了一个可以产生失真波形的放大器非线性失真研究装置，同时具有测定波形谐波电压值与近似总失真系数的功能。作品以 STM32 单片机和 FPGA 为主控制器，STM32 控制 AD9959 DDS 模块，产生频率 1KHz、峰峰值 20mV 稳定正弦波，并控制一个多路继电器，使三极管放大电路输出五种失真波形。波形先经过以 OPA188 为主的信号调理电路，由 STM32 控制模块进行信号跟随或信号放大。再送入 ADS8505 模块对信号进行模数转换。FPGA 先对采样倒的波形数据缓存，再通过并行传输送入 STM32。STM32 对接受到的波形数据进行 FFT，获得信号频谱数据，读取各个谐波分量电压值并进行总谐波失真系数计算。最后将各个谐波分量电压值、总谐波失真系数、电压波形与数字频谱图进行屏幕打印。

关键词：放大器非线性失真分析；幅频特性；FFT；FPGA

一、方案比较（论证）

1、采集测量与微处理器方案比较与论证

方案一：使用 STM32F407 内置 ADC 采集晶体管放大器输出信号，并选择 STM32F407 作为微处理器，进行 FFT 处理获取谐波分量。这种方案的优点是系统构成简单，所需硬件资源较少，但 STM32F407 的内置 ADC 只能测量 0V~3.3V 的输入电压，并且只有 12 位精度，无法保证精确测量，对信号 FFT 处理的影响较大。同时，FFT 要求采样频率为 2 的指数，STM32 无法保证稳定精确的采样频率，这也会影响 FFT 的结果，最终影响总谐波失真的准确测量。

方案二：通过 ADS8505 采集晶体管放大器输出信号，并使用 FPGA 缓存数据。STM32 与 FPGA 进行并行传输获取缓存数据，计算 FFT 获取谐波分量。ADS8505 的电压测量范围为-10V~+10V，具有 250KHz 采样率，精度高达 16 位。通过 FPGA 设置精确时钟，可保证采样频率。同时，FPGA 与 STM32 的并行传输，确保数据快速准确传输至 STM32F407 并进行 FFT 处理。此方案可以提高输入电压范围，大大提高输入信号 FFT 的精确度。

综上所述，本系统采用方案二。

2、共射放大器级联方案比较与论证

方案一：采用单级共射放大器。根据理论计算及仿真分析，若采用 S9013H 三极管，负载为$100K\Omega$时的电压增益约为$120V/V$，可满足题目对于增益的要求，但因无法使三极管同时工作在饱和区、截止区而产生双向失真，并且在后级功率放大器产生交越失真时，由于增益过低也无法满足题目的输出幅值要求。

方案二：采用两级共射放大器级联。若采用两级阻容耦合共射放大器，可通过控制第一级放大器增益产生一个大信号使第二级三极管产生双向失真，也可以轻松控制整个放大器的电压增益，使输出幅值达到题目要求。

综上所述，本系统采用方案二。

3、AB 类功率放大器克服交越失真方案比较与论证

方案一：使用电阻压降产生偏置电压，使功率放大管处于微导通的状态，可以有效的克服交越失真，虽然电路简单，但是电阻压降产生的偏置电压不稳定可能会造成功率放大器的总谐波失真变大。

方案二：使用V_{be}扩大电路产生偏置电压。V_{be}扩大电路可以通过改变偏置电阻的比值，从而改变功率放大管的偏压值，但电路结构较为复杂。

方案三：使用二极管压降产生偏置电压不但有着和电阻压降偏置同样简单的电路，而且偏置电压较为稳定，且由于二极管的正向导通特性，无需复杂的设计即可产生足够的偏置电压。

综上所述，本系统采用方案三。

4、ADC 信号调理方案比较与论证

方案一：仅使用运放跟随器。使用一颗精密运放进行跟随可以实现阻抗变换，减小后级对前级电路的影响，但是测量$20mV_{pp}$的输入信号时，由于信号幅值过于微小，ADS8505 无法准确转换。

方案二：使用程控增益放大器。程控增益放大器可以轻松改变放大器增益，但电路较为复杂且多数为高频电路设计，低频性能不佳，若要用于本题则需要重新计算有关参数。

方案三：使用继电器切换同相放大器。使用继电器切换由运放组成的同相放大电路，不仅可以实现阻抗变换的功能，而且可以在测量微小信号时使其放大一定倍数，使 ADS8505 能够准确转换。

综上所述，本系统采用方案三。

二、理论分析与计算

1．共射放大电路静态工作点估算及交流电压放大倍数

共射放大电路如图 1 所示。在确定该电路静态工作点时，为保证 Q 点的稳定，估算时选取：静态时R_{b2}流过的电流$I_b = (5\sim10)I_{BQ}$,$V_{BQ} = (5\sim10)U_{BEQ}$。

计算公式如下：

$$V_{BQ} \approx \frac{R_{b2}}{R_{b1}+R_{b2}}V_{CC} \tag{1}$$

$$I_{CQ} \approx \frac{V_{BQ}-U_{BEQ}}{R_e} \tag{2}$$

$$U_{CEQ} = V_{CC} - I_{CQ}(R_C + R_e) \tag{3}$$

放大器交流放大倍数：

$$A_V = -\frac{\beta(R_L//R_C)}{r_{be}} \tag{4}$$

其中，r_{be}为三极管输入电阻。其值为$r_{be} = r_{bb'} + (1+\beta)\frac{U_Q}{I_{EQ}}$

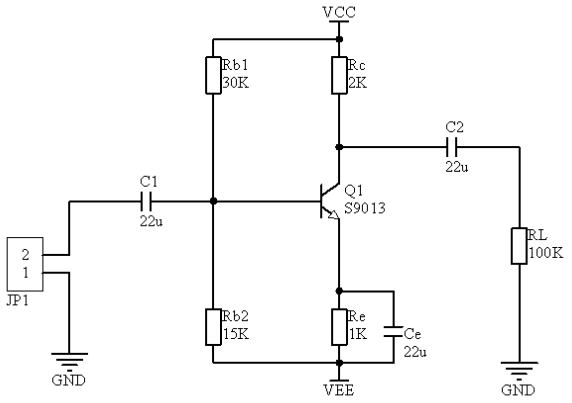

图 1 分压偏置式共射放大电路

其中，VCC=5V，VEE=-5V，β取 200，I_{CQ}取 1.8mA，U_{BEQ}取 0.64V，则V_{BQ}=-2.5V，I_{BQ}=8.85μA，令R_{b2}=15KΩ，则R_{b1}≈43KΩ；令R_e=1$K\Omega$，则$R_c \approx 2K\Omega$。计算$Av \approx 129V/V$。实验发现有轻微削顶失真，故减小 Rb1 至30$K\Omega$。

2．失真波形理论分析

1）顶部失真：三极管静态工作点过高，静态工作电流过小，管子进入截止区，产生截止失真。

2）底部失真：三极管静态工作点过低，静态工作电流过大，管子进入饱和区，产生饱和失真。

3）双向失真：放大器的输入信号幅值过高，导致信号正半周时管子进入截止区，产生截止失真；信号负半周时管子进入饱和区，产生饱和失真，这两种失真叠加就产生了双向失真。

4）交越失真：推挽式输出通常使用 NPN 型管和 PNP 型管，两管的基级和发射极相互连接在一起，连接负载电阻。当有信号输入时，两个管子轮流导通。若输入信号低于两管导通电压，两管都截止，负载上无电流通过，因此出现一段死区，即交越失真。

3、总谐波失真系数计算

总谐波失真的测量：总谐波失真是指用信号源输入时，输出信号比输入信号多出的额外谐波成分，其计算方法为：

$$THD = \frac{\sqrt{U_{o2}^2+U_{o3}^2+U_{o4}^2+L+U_{on}^2}}{U_{o1}} \times 100\% \quad (5)$$

三、电路与程序设计

1.硬件电路设计

（1）硬件总体框图

总体框图如图 2 所示。以 STM32F407 单片机为主控制器，通过外部输入源或 AD9959 作为信号源，产生 1KHz，20mV 正弦波，正弦波首先经过两级共射放大器，单片机可以通过控制电路中继电器，控制一级放大器增益与二级放大器输入电压直流偏置，分别产生无失真正弦波、顶部失真、底部失真与双向失真波形。之后进入三极管推挽电路，单片机控制推挽电路产生交越失真信号。最后进入 ADC 信号调理电路，进行信号跟随。ADS8505 以 64KHz 采样率进行模拟电路输出信号采样，由 FPGA 缓存后通过并行通信传输至单片机，单片机进行 FFT 处理，获取各谐波分量电压值与近似总谐波失真值。最后在 LCD 屏幕上显示。

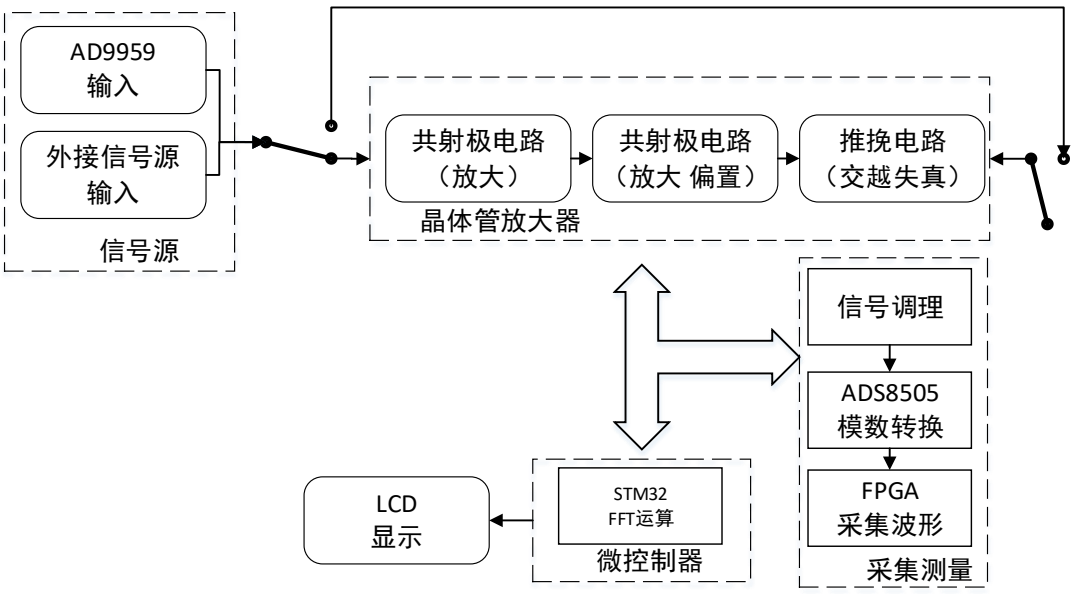

图 2 硬件总体框图

（2）程控放大器电路

本电路由两级三极管共射放大器和一级 AB 类三极管功率放大器组成，通过继电器切换电阻和短路偏置二极管的方式，实现了程控产生四种失真波形的功能。（如图 3 所示）

1）正常情况：继电器 K1，K3 吸合，其余继电器释放，Rp1 分压减小第二级输入幅值，第二级放大不会发生失真，基极偏置电阻 Rb1，Rb2，Rb3，Rb4 为两个晶体管提供正常偏置，后级功放管也处于微导通状态，无交越失真，电路正常工作。

2)顶部失真：继电器 K2，K4 吸合，其余继电器释放， Rp2 阻值较小，使输入第二级共射放大器的信号幅值增大，同时 Rp3 调整至比 Rb4 大的某一阻值，使第二级放大器工作在截止区，波形出现顶部失真。

3)底部失真：继电器 K1，K5 吸合，其余继电器释放，其中 Rp4 调整至比 Rb4 小的某一阻值，使第二级放大器工作在饱和区，波形出现底部失真。

4)双向失真：继电器 K2，K3 吸合，其余继电器释放，其中 Rp2 调整成较小阻值，使输入第二级共射放大器的信号幅值增大，使第二级放大器工作在饱和区，截止区而产生双向失真。

5)交越失真：继电器 K1，K6，K7 吸合，其余继电器释放，前两级共射放大器正常工作，末级功放由于二极管被短路，功放管失去直流偏置，信号低于管子导通电压时管子无法导通，产生交越失真。

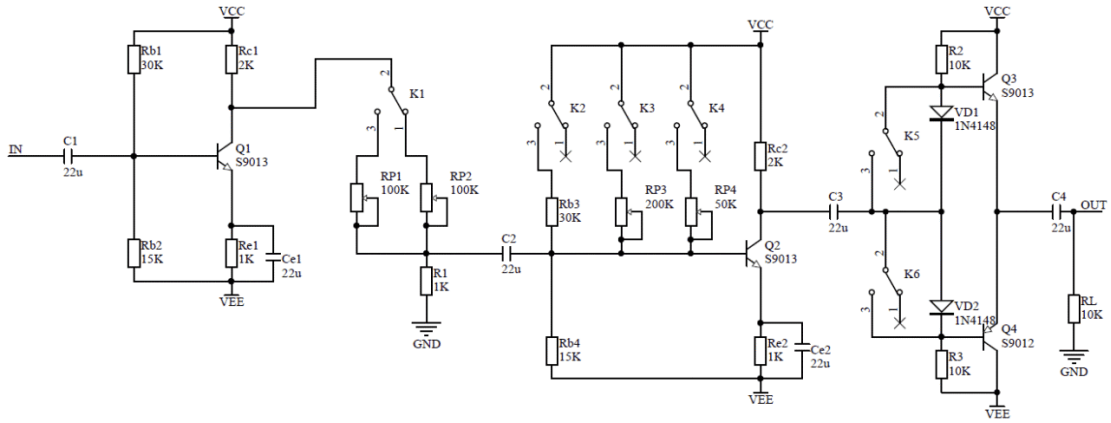

图 3 程控放大器电路原理图

(3)电源方案设计

系统电源方案分为模拟电源与数字电源两部分，模拟电源采用线性稳压器78XX、79XX 系列，为系统内放大器电路与信号调理电路供电。数字电源采用开关电源 TPS5430 系列，为系统内 STM32 与 FPGA 供电。

2.程序设计

STM32 控制 ADS9959 输出 1KHZ、20mVpp 的正弦波作为输入信号；FPGA 控制 ADS8505 以 64KSPS 速率采样，将得到的 ADC 数据存入内部的 RAM 中。当 STM32 通过并口传输协议向 FPGA 发送数据请求时，FPGA 发送 ADC 采集到 256 个电压值发送给 STM32。STM32 根据得到的数据进行点数为 256 的 FFT 变换，得到各个谐波分量的频谱信息，经过计算得到“总谐波失真”的近似值，最终将得到的结果显示在 LCD 上。软件系统流程图如图 4 所示，“总谐波失真”的近似值计算流程图如图 5 所示。

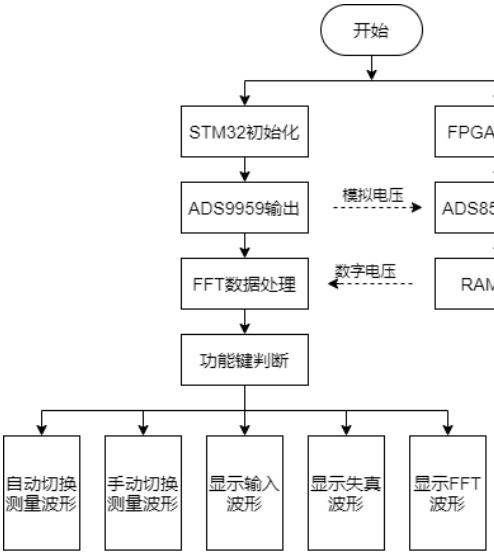

图 4 软件系统流程图

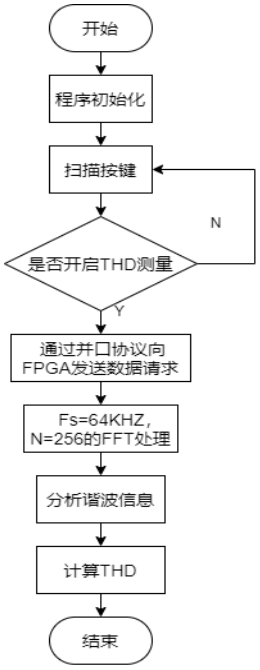

图 5“总谐波失真”的近似值计算流程图

四、测试方案与测试结果

1、系统测试方案

采用外接信号源，通过硬件电路产生无失真或各种失真类型正弦波。用

ADS8505 采样输出电压，通过并行通信将数据传递至 STM32，STM32 进行 FFT 计算得到幅频特性曲线，并获取各谐波电压幅值，计算总谐波失真。

表 1 测试仪器列表

序号	仪器类别	仪器型号	数量	性能参数
1	信号源	DG822	1	10HZ-35MHZ
2	示波器	MSO5102	1	500Mhz
3	学生电源	DP832	1	30V/3A*2
4	功率分析仪	PA6000H	1	可测量总谐波失真

2、测试结果与分析

（1）三极管放大器电路输出波形测试。

输入信号为 20mVpp，1KHz 正弦信号，测量输出电压波形，与电压波形幅值。（无失真及各种失真波形图片见附录图片 6、7、8、9、10、11）

表 2 各失真波形峰峰值电压

波形种类	峰峰值电压
无明显失真电压波形	3.120V
顶部失真电压波形	4.736V
底部失真电压波形	2.908V
双向失真电压波形	6.232V
小交越失真电压波形	2.654V
大交越失真电压波形	2.512V

根据实验数据，放大器电路各类失真输出波形符合题目要求，输出电压峰峰值都达到了 2V 以上，完全满足题目要求。

（2）采集测量模块 THD 误差测量。

输入信号为 20mVpp，1KHz 正弦信号，通过三极管放大电路输出各类失真波形，与功率分析仪 PA6000H 测量 THD 值对照。结果如表 3 所示。

表 3 各失真波形 THD 值测试

输出波类型	功率分析仪 THD 值	实际测量 THD 值	误差
无明显失真正弦波	4.761%	4.804%	0.903%
顶部失真正弦波	60.941%	61.325%	0.630%

底部失真正弦波	14.751%	14.702%	-0.332%
双向失真正弦波	24.785%	25.187%	1.621%
小交越失真正弦波	10.743%	10.468%	-2.560%
大交越失真正弦波	22.536%	22.500%	-0.160%
标准方波	38.889%	39.050%	0.414%
标准三角波	11.782%	11．781%	-0.009%

根据实验数据，采集测量模块测量 THD 值与实验仪器测量 THD 值误差保持在 3%以内，有着比较高的采集测量精度。

五、总结

本作品设计了一个放大器非线性失真研究装置，实现了信号发生，波形放大、程控输出多种失真波形与测量信号近似总谐波失真数值的功能。为了实现波形放大与输出 5 种失真波形，必须准确计算放大电路电压增益与偏置电压，设计放大器电路参数。与此同时，我们使用 FPGA 稳定的时钟晶振保证采样频率为 0.25kHz 的倍数，加快了采样速率，提高了 FFT 计算精度，保证了计算各谐波电压值的准确性。

附录

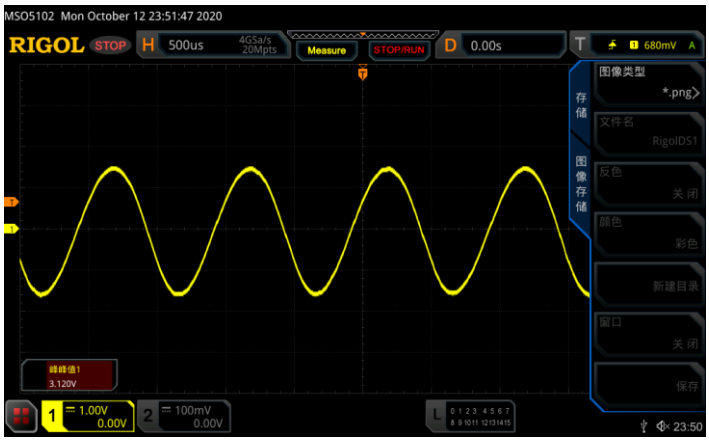

图 6 无失真正弦波

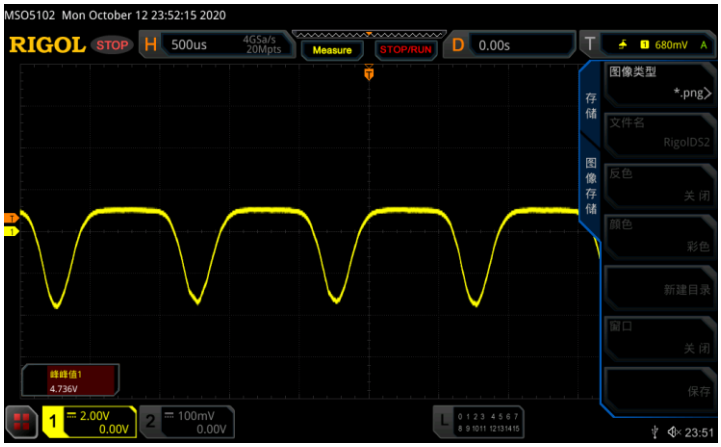

图 7 顶部失真

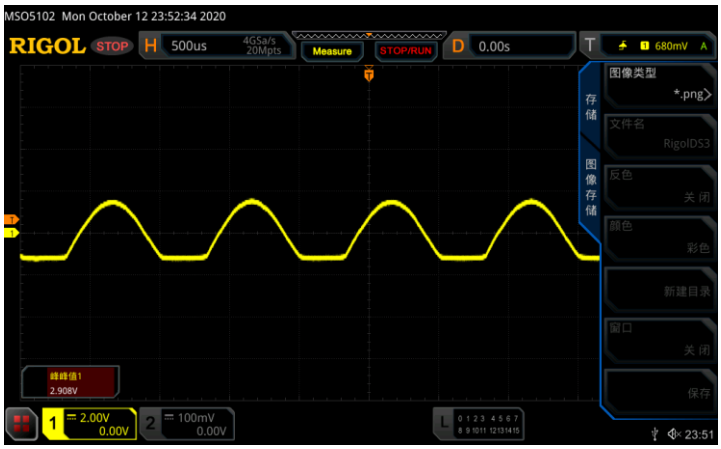

图 8 底部失真

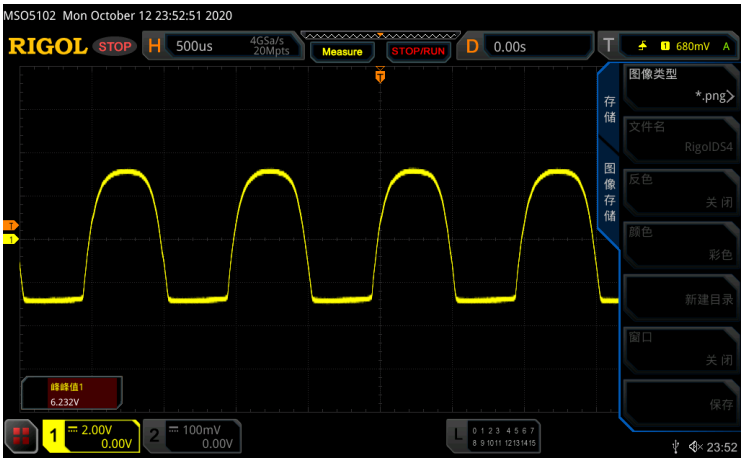

图 9 双向失真

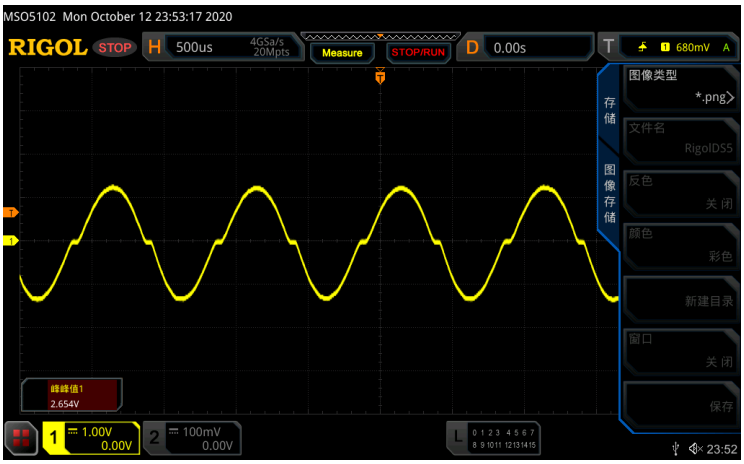

图 10 小交越失真

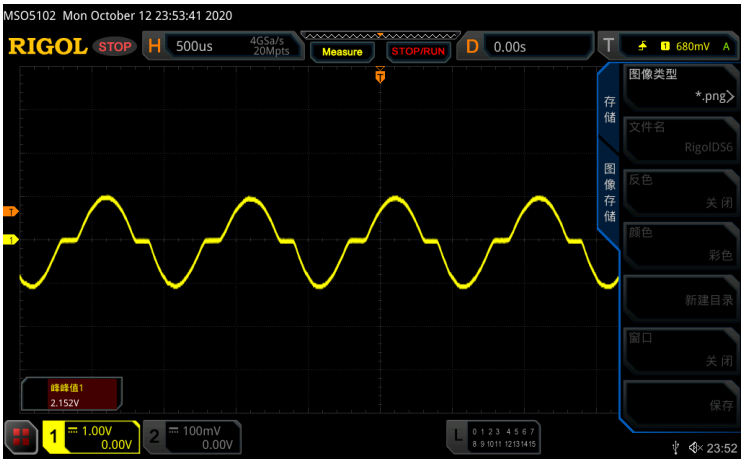

图 11 大交越失真

```
通信协议：
void read_all_data(void)
{
    u16 i =0 ;
    int k = 0;
    CMBUS_START_H;
    for(i=0; i<ADC_DATA_NUM_MAX; i++)
    {
        CMBUS_START_L;
      for (k = 0; k < 100; ++k);                    //延时
        ADC_Cache[i] = ADC_Read_Data();          //数据存到16字节缓冲区
       CMBUS_START_H;
       for (k = 0; k < 100; ++k);                //延时
    }
}
数据处理：
void led0_task(void *p_arg)
{
    OS_ERR err;
    u16 i;
    arm_cfft_radix2_init_f32(&scfft,ADC_DATA_NUM_MAX,0,1);  //初始化scfft结构体，设定FFT相关参数
    while(1)
    {
        LED0 = !LED0;
            //*********************读取数据*********************//
      read_all_data();
            //*********************数据处理*********************//
            for(i=0;i<ADC_DATA_NUM_MAX;i++)   //转换为实际电压值
                {
                    if(ADC_Cache[i] > 32767)
                    {
                        fft_inputbuf[2*i] = (ADC_Cache[i]- 65536)*10.0f/32768;
                    }
                    else
                    {
                        fft_inputbuf[2*i] = ADC_Cache[i]*10.0f/32768;
                    }
```

图 12　STM32 代码

```
                fft_inputbuf[2*i+1]=0.0;
            }
            if (normal_or_fft == 0)         //是否显示正常波形
            {
                wave_indicate_flag  = 0;
              wave_ok_flag = 1;
                while (wave_indicate_flag == 0)
                {
                    OSTimeDlyHMSM(0,0,0,10,OS_OPT_TIME_PERIODIC,&err);
                };
            }
          //**********************FFT运算************************//
            arm_cfft_radix2_f32(&scfft,fft_inputbuf);                   //FFT计算（基2）
            arm_cmplx_mag_f32(fft_inputbuf,fft_outputbuf,ADC_DATA_NUM_MAX); //把运算结果复数求模得幅值
            jibo_temp = fft_outputbuf[1];
            for(i=2;i<31;i++)
            {
                if(fft_outputbuf[i]>jibo_temp)
                {
                    jibo_xiabiao = i;
                  jibo_temp = fft_outputbuf[i];
                }
            }
          //*********************计算THD*********************//
            xiebo1 = fft_outputbuf[jibo_xiabiao];
            xiebo2 = fft_outputbuf[jibo_xiabiao*2];
            xiebo3 = fft_outputbuf[jibo_xiabiao*3];
            xiebo4 = fft_outputbuf[jibo_xiabiao*4];
            xiebo5 = fft_outputbuf[jibo_xiabiao*5];
            xiebo6 = fft_outputbuf[jibo_xiabiao*6];
            xiebo7 = fft_outputbuf[jibo_xiabiao*7];
            THD3 = sqrt(xiebo3*xiebo3+xiebo2*xiebo2)/xiebo1*100;
            THD5 = sqrt(xiebo5*xiebo5+xiebo4*xiebo4+xiebo3*xiebo3+xiebo2*xiebo2)/xiebo1*100;
            THD7 = sqrt(xiebo7*xiebo7+xiebo6*xiebo6+xiebo5*xiebo5+xiebo4*xiebo4+xiebo3*xiebo3+xiebo2*xiebo2)/xiebo1*100;
        fft_ok_flag = 1;
            OSTimeDlyHMSM(0,0,0,250,OS_OPT_TIME_PERIODIC,&err);//延时500ms
    }
```

图 13　STM32 代码

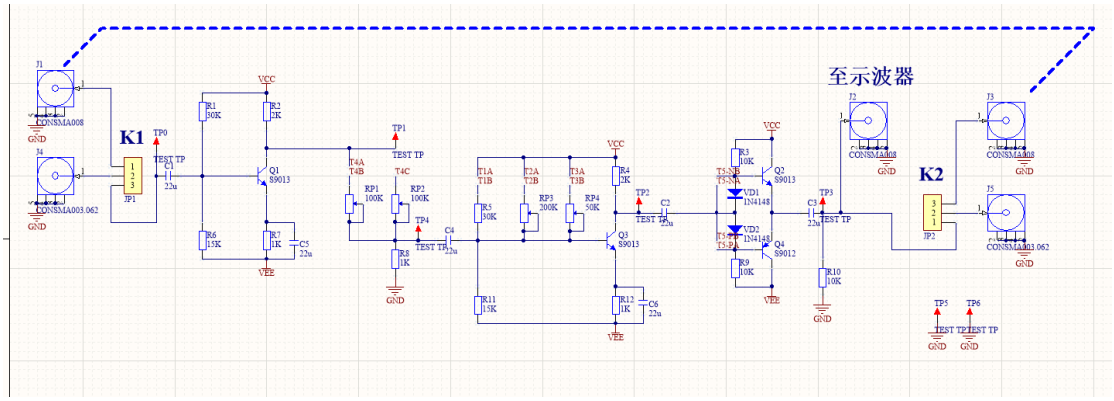

图 14 放大器电路原理图

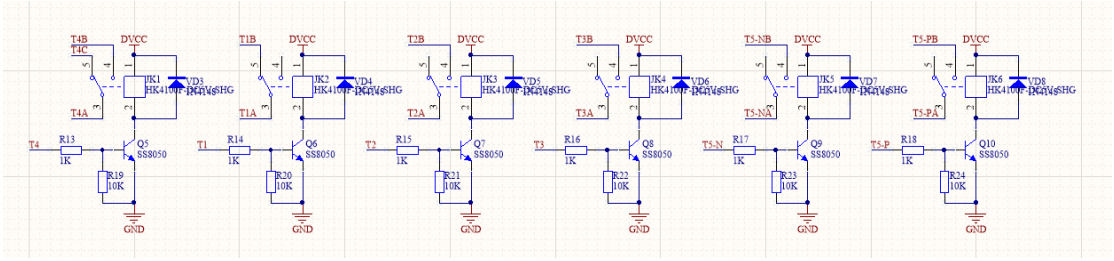

图 15 放大器电路原理图

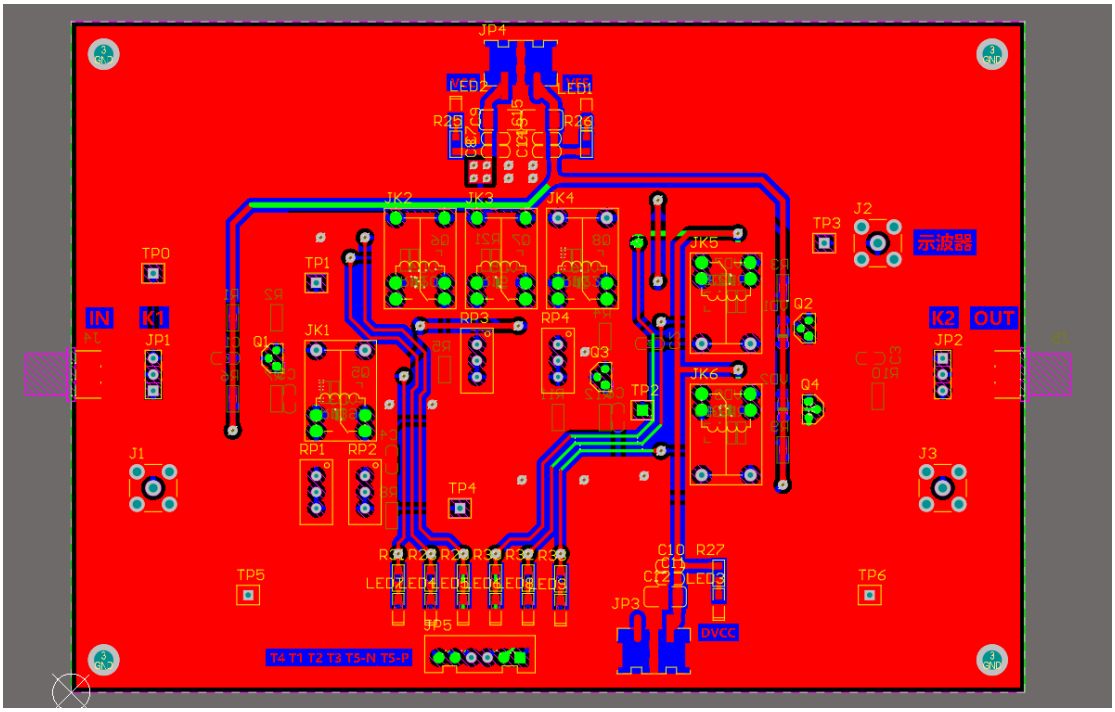

图 16 放大器电路 PCB（正）

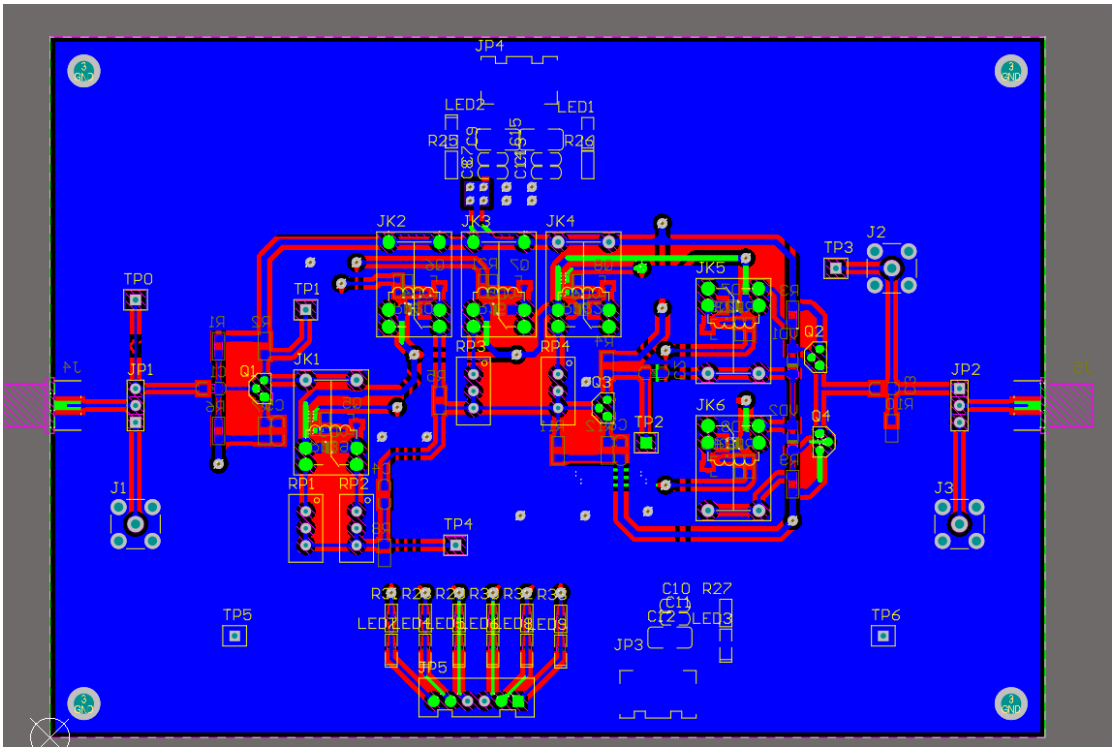

图 17 放大器电路 PCB（反）

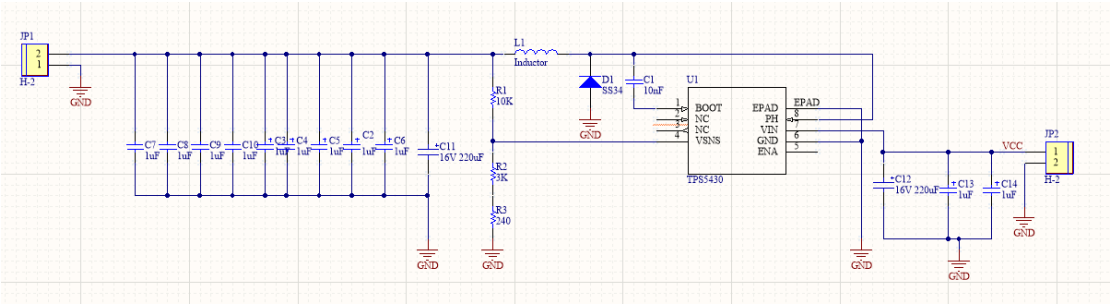

图 18 TPS5430 原理图

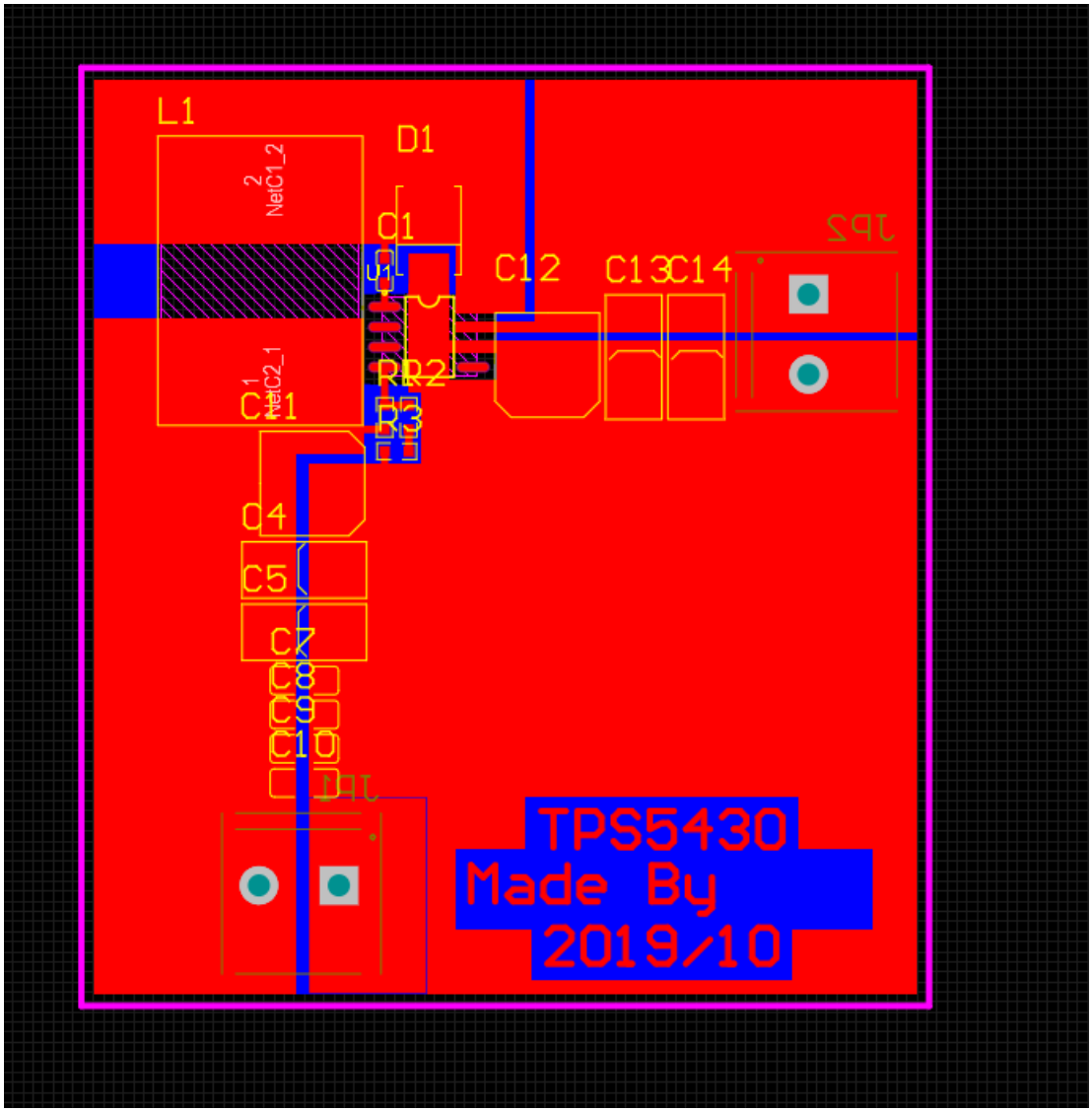

图 19 TPS5430 电路

```
65  assign LED_light=~full_buff_flag;
66  always@(posedge sys_clk or negedge rst_n) begin
67      if (!rst_n)
68          begin
69              address<= 10'b0;
70              full_buff_flag<=1'b0;
71          end
72      else if (full_buff_flag)
73          if (address == 10'd1023)
74              begin
75                  full_buff_flag<=1'b0;
76                  address<=10'b0;
77              end
78          else if (address == 10'b0) begin
79              RAM_cnt <= RAM_cnt + 2'b1;
80              if (RAM_cnt == 2'd3) begin
81                  address <= address +10'b1;
82                  RAM_cnt <= 2'b0;
83              end
84          end
85          else begin
86              if (cs_done)    //cs为低电平
87              begin
88                  address <= address +10'b1;
89                  end
90              end
91      else begin
92          if(address == 10'd1023)
93              begin
94                  address<= 10'b0;
95                  wren <= 1'b0;
96                  full_buff_flag<=1'b1;
97              end
98          else if (address == 10'b0) begin
99                  RAM_cnt <= RAM_cnt + 2'b1;
100                 wren <= 1'b1;
101             if (RAM_cnt == 2'd3) begin
```

图 20 FPGA 代码

```
101             if (RAM_cnt == 2'd3) begin
102                 address <= address +10'b1;
103                 // data <= address;
104                 RAM_cnt <= 2'b0;
105             end
106         end
107         else  begin
108         if (done_sig)
109             begin
110                 address <= address +10'b1;
111             end
112         end
113     end
114 end
115 
116 wire    wren_wire;
117 wire    [9:0]   address_wire;
118 assign wren_wire = wren;
119 assign address_wire = address;
120 wire [15:0] RAM_out;
121 assign Data_out = (full_buff_flag)?RAM_out:15'b0;
122 //assign Data_out = RAM_out;
123 ads8505  U_ads8505(
124     .clk                (sys_clk),
125    .rst_n          (rst_n),
126     .adc_rc           (adc_rc)
127 );
128 
129 RAM1 U_RAM1(
130     .address        (address_wire),
131     .data           (Data_in),
132     .inclock        (sys_clk),
133     .wren           (wren_wire),
134     .q              (RAM_out)
135     );
136 
137 endmodule
```

图 21 FPGA 代码