
远程幅频特性测试装置设计报告

摘要：本设计以 FPGA 与单片机为核心完成了远程幅频特性测试装置的构建。设计主要包括 DDS 集成模块、放大器模块、幅频特性测试装置、WIFI 模块以及有线信道模块。系统采用 DDS 集成模块将其产生的扫频信号通过放大器模块，经放大衰减后得到符合要求的信号。最后输入幅频特性测试装置进行测量，并在示波器上显示放大器输出信号的幅频特性。

关键词：幅频特性；放大器；WIFI 传输

Abstract: The design of FPGA and microcontroller as the core to complete the remote amplitude characteristics of the test device to build. The design includes DDS integrated module, amplifier module, amplitude-frequency characteristic test device, WIFI module and cable channel module. The system uses the DDS integration module to generate the swept signal through the amplifier module, after amplification attenuation to meet the requirements of the signal. Finally, input amplitude frequency characteristic test device to measure, and in the oscilloscope display amplifier output signal amplitude characteristics.

Key words: Amplitude-frequency characteristics; Amplifier; WIFI transmission

目录

一、方案论证与比较	1
1.1 信号源方案选择	1
1.2 控制平台选择	2
1.3 放大器方案选择	2
1.4 上位机设计方案选择	3
二、理论分析与计算	3
2.1 系统原理	3
2.2 放大器设计	4
2.3 DDS 信号源设计	5
三、电路设计	6
3.1 DDS 信号源	6
3.2 放大器	6
3.3 检波模块	7
3.4 有线信道模块	8
四、软件程序设计	8
4.1 单片机控制和显示设计	8
4.2 FPGA 程序设计	9
4.3 局域网配置和上位机设计	10
五、测试方案与结果	14
5.1 测试仪器	14
5.2 测试方案	14
5.3 测试结果	15
六、设计实现	17
七、结论	17
参考文献	18
附录	18
附录 1 AD9865 信号源电路	18
附录 2 VCA810 放大器电路	20
附录 3 AD8307 检波电路	21
附录 4 FPGA 下位机代码	22
附录 5 上位机 QT 界面代码	25

一、方案论证与比较

系统主要由四个模块组成，直接数字频率合成 DDS（Direct Digital Synthesizer）信号源模块，放大器模块，有线信道模块，转换电路 WiFi（Wireless Fidelity）模块，幅频特性测试模块，上位机模块。系统总体框图如图 1 所示。

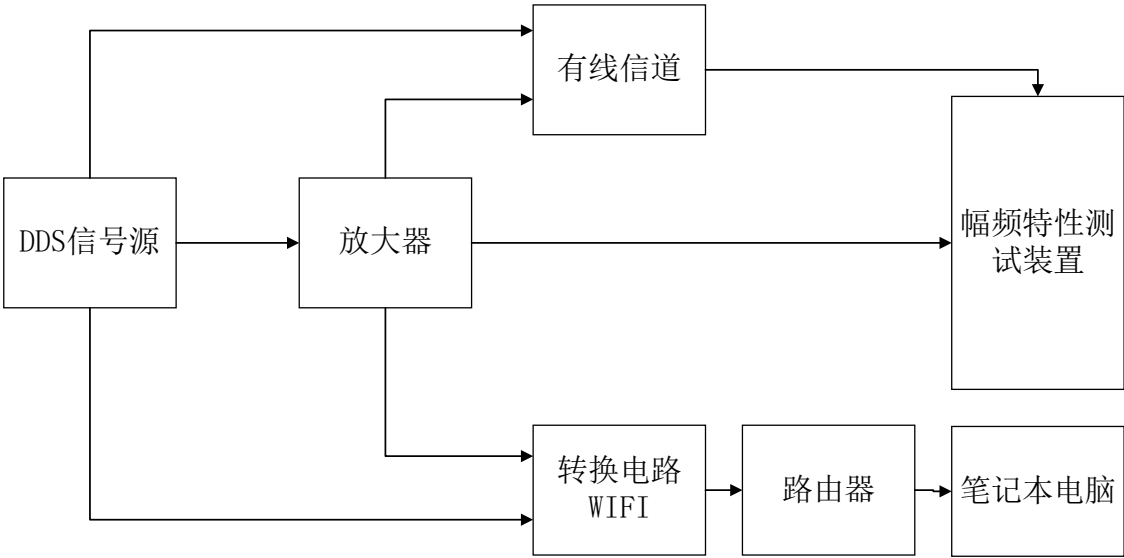


图 1 系统总体框图

下面就关键模块的设计方案进行讨论与分析。

1.1 信号源方案选择

方案一：采用锁相环间接频率合成方案。锁相环频率合成在一定程度上解决了既要求频率稳定精确，又要求频率在较大范围可调的矛盾。但输出频率易受可变频率范围的影响，输出频率相对较窄，不能满足题目 1MHz-40MHz 的高频要求。

方案二：采用高集成度的 AD9854 数字合成器，它可以输出两路正交的本地载波，高达 300MHz 的系统时钟，内部有 4 到 20 的可编程时钟倍频器，两个 48 位频率控制字寄存器，能够实现很高的频率分辨率，带有 100M 的 8 位并行数据传输口或 10MHz 的串行数据传输口完全满足题目要求。

综合考虑，采用 AD9854 DDS 芯片可实现题目对于信号源的要求。本设计选

用方案二。

1.2 控制平台选择

方案一：采用 FPGA 或 CPLD 进行控制

采用现场可编程逻辑器件 FPGA 进行控制。利用板上丰富的引脚端口，可同时对 DDS 信号源模块，人机交互显示以及键盘等的控制。同时具有频率稳定，精度高等优点。但考虑 FPGA 本身并行执行的特点，对于 DDS 信号源模块以及显示的控制不够灵活。

方案二：采用单片机进行控制

采用 STC12 单片机进行控制。STC12 是增强型 8051 内核 CPU，1T 是传统的 51 的 12 倍速度，指令代码完全兼容传统 8051，是一种高速高效率的 8051 兼容单片机。相比 FPGA，具有更灵活，精巧的控制方案，利用板载引脚，完全可以实现控制命令输入，以及 DDS 扫频源的控制，其性价比高。但其板载晶振频率低，无法实现高频测量。

方案三：采用单片机和 FPGA 进行控制

采用 Intel FPGA 加 STC89C51 单片机进行控制。STC89C51 单片机具有控制灵活，成本低廉的优点。同时 FPGA 处理速度快，适用于信号源输出的高频测量。

综合考虑，用单片机作为控制单元，FPGA 作为频率测试单元能够满足题目要求，因此选用方案三。

1.3 放大器方案选择

方案一：采用多级运放级联设计放大器。但多级运放调试复杂，放大器的稳定性差，且难以实现放大器的增益的连续可调。

方案二：采用 VCA810 可变高增益设计放大电路。VCA810 的增益能够在 -40dB-40dB 连续可调，且带宽能够基本满足放大器的要求，电路设计简单。

方案三：采用 AD603 压控放大器设计放大器，AD603 能提供由直流到 30MHz 的工作带宽，实际工作时可提供 20dB 以上的增益，配合前、后级的放大器和电阻

衰减网络就可以实现 60dB 以上的增益调节，电路较为复杂。

综合考虑，方案二精度高，且较容易，故选用该方案。

1.4 上位机设计方案选择

方案一：采用 VB 语言进行上位机界面设计，简化了 Windows 程序界面设计工作，只需要极少量代码就能实现 Windows 应用程序界面。但开发周期相对较长，且 VB 下界面编程技巧复杂，难以设计和实现既符合一般标准又具有特色的界面。

方案二：采用 LabView 进行上位机界面设计。LabView 是 NI 公司推出的虚拟仪器软件开发平台，使用编译型图形化编程语言 G 语言，同时通过模块化编程实现各种复杂功能。但程序修改复杂，移植性差，且开发周期长。

方案三：采用 Qt 进行图形用户界面设计。作为跨平台的 C++ 应用程序开发框架，既可以开发 GUI 程序，也可用于控制台工具和服务器，功能强大。可同时支持多平台的界面设计。开发周期短，实现简单。

综合考虑，方案三适用强，开发周期短，效率高，故选用该方案。

二、理论分析与计算

2.1 系统原理

为了满足题目要求，设计核心应放在信号源，放大器的参数设计上。在芯片的选型上，为了满足系统 $1\text{MHz} \sim 40\text{MHz}$ 的频带的要求，DDS 器件采用具有高速数字电路和高速 D/A 转换技术的 AD9854 芯片。输入信号为有效值 10mV 时，输出峰峰值大于 200mV，通过级间阻抗匹配后衰减二分之一， $V_{pp} > 100\text{mV}$ 。两个 48 位的可编程频率寄存器；两个 14 位的可编程相位偏移寄存器；双 12 位可编程幅度控制寄存器和键控可编程幅度渐变开关功能；32 位控制寄存器，便于并行传输。

2.2 放大器设计

VCA810 的 3 号引脚增益控制端的参考电位是为 $0V \sim (-2)V$ 可实现 $-40dB \sim 40dB$ 增益可调的。器件增益控制电压在 $\pm 5V$ 电源下工作，在 $-2V$ 输入下，将 $0V$ 输入的 $-40dB$ 增益调整至 $-2V$ 输入时的 $40dB$ 。该 $40dB/V$ 增益控制精度在 $\pm 1.5dB$ 以内。

增益转换关系：

$$G(dB) = -40 \times (V_c + 1) dB \quad (1)$$

(V_c 为增益引脚控制端电位，范围为 $0V \sim (-2)V$)

满足增益可调条件下，计算得： $V_c = (-1V) - (-2V)$ 。

通过 lm358 直流电压增益电路控制 vca810 增益参考电位。通过电位器获得电压通过电压跟随器输入到反相器中获得负压。

$$\text{电压跟随器: } U_{o1} = U_{i1} \quad (2)$$

$$\text{反相器: } A_u = \frac{U_{o2}}{U_{o1}} = -1 \quad A_u = \frac{-R_f}{R_5} = -1 \quad (3)$$

$R_f = R_5 = 1K$ 可实现反向电压效果

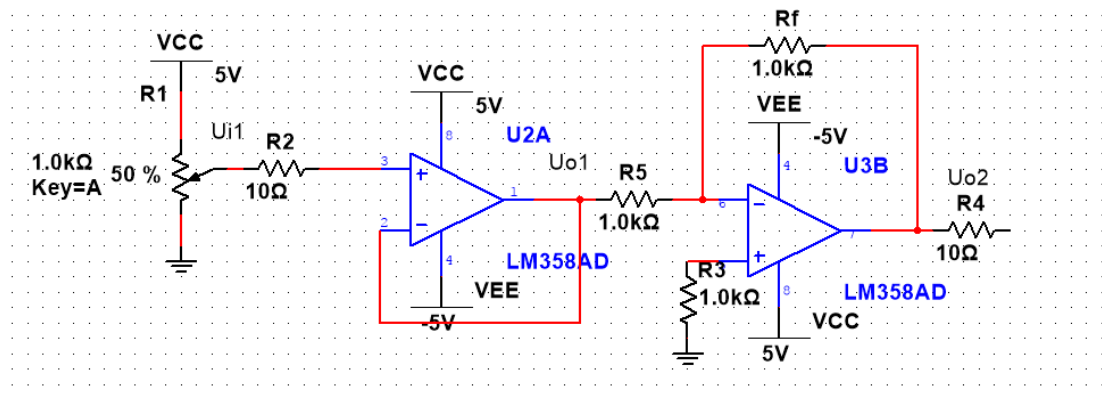


图 2 放大器原理图

2.3 DDS 信号源设计

DDS 信号源采用 AD9854 作为信号源。时钟源为高稳定度的晶体振荡器，其输出用于提供 DDS 中各器件同步工作，频率控制字和相位控制字用于设定输出波形的频率和相位，高速 DAC 用于波形的 D/A 转换，LPF 用于滤除波形中的杂散噪声信号。

相位累加器一部分是 DDS 系统的最核心部分，相位累加器由 N 位加法器和 N 位相位寄存器构成，其环节是典型的反馈电路，每来一个时钟脉冲，累加器就将频率控制字 K 与相位寄存器输出的累加相位数据相加，把相加的结果送至相位寄存器数据输入端，相位寄存器将累加器在上一个时钟作用后所产生的新相位数据反馈到累加器的输入端，以使累加器在下一个时钟的作用下继续与频率控制数据相加。这样，相位累加器在参考时钟的作用下，进行线性相位相加。当累加器累加到满量时就会产生一次溢出，完成一个周期性的动作，这个周期就是 DDS 合成信号的一个频率周期。累加器的溢出频率就是 DDS 输出的信号频率，相位调制器这一部分是接收相位累加器的输入，在这里加上一个相位偏移值，主要用于信号的相位调制。波形存储器产生任意波形。

数模（D/A）转换和滤波部分中，将数字量形式的波形幅值转换成所要求合成频率的模拟量形式的信号，低通滤波器用于衰减和滤除不需要的取样分量以输出频谱纯净的波形信号。

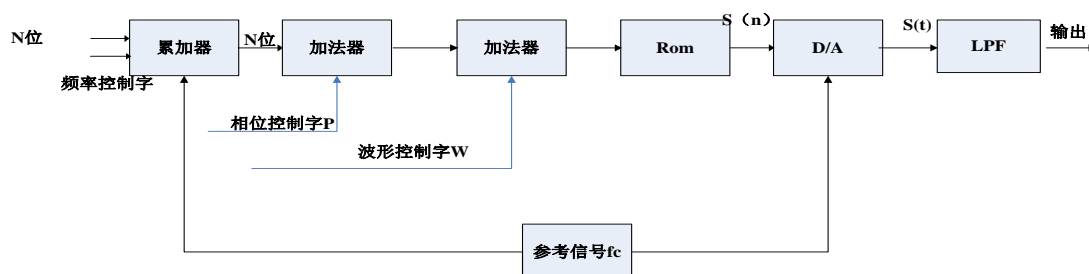


图 3 AD9854 内部工作原理框图

三、电路设计

3.1 DDS 信号源

AD9854 需要 3.3V 供电，电源稳压模块提供 3.3V. 工作频率可达 300MHz，满足输出频率范围：1MHz - 40MHz；步进：1MHz；AD9854 芯片内有 5 种模式: Single-Tone、Un2rampedFSK、RampedFSK、Chirp 和 BPSK. 调制功能丰富, 可实现复杂的数字调制和模拟调制等。

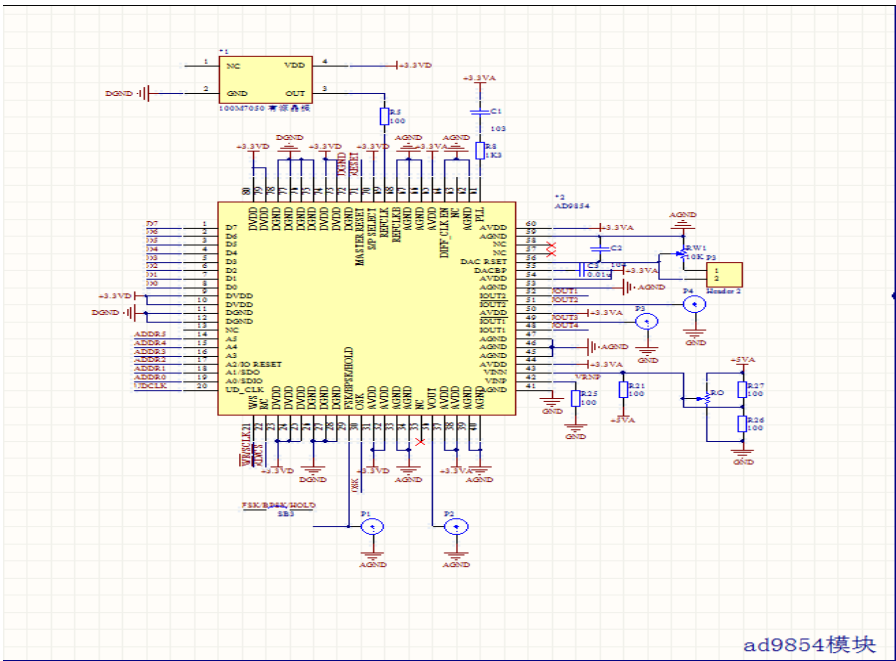


图 4 AD9854 放大电路原理图

3.2 放大器

题目要求放大器的输入阻抗和输出负载为 600Ω ，经实际调试在 VCA810 输入端并联 600Ω 电阻时，输入阻抗为 500Ω ，经过调整电阻参数得到阻值为 650Ω 时输入阻抗为 600Ω 。在输出端并联 600Ω 为负载电阻，在输出端加一个 $0.1\mu f$ 电容滤除直流分量，整体结构满足要求。后级由 LM358 建立电压跟随器和反向电路，输出电压控制增益可调，最后测试结果为输出电压负压可调，满足放大电路电压增

益可调的范围。

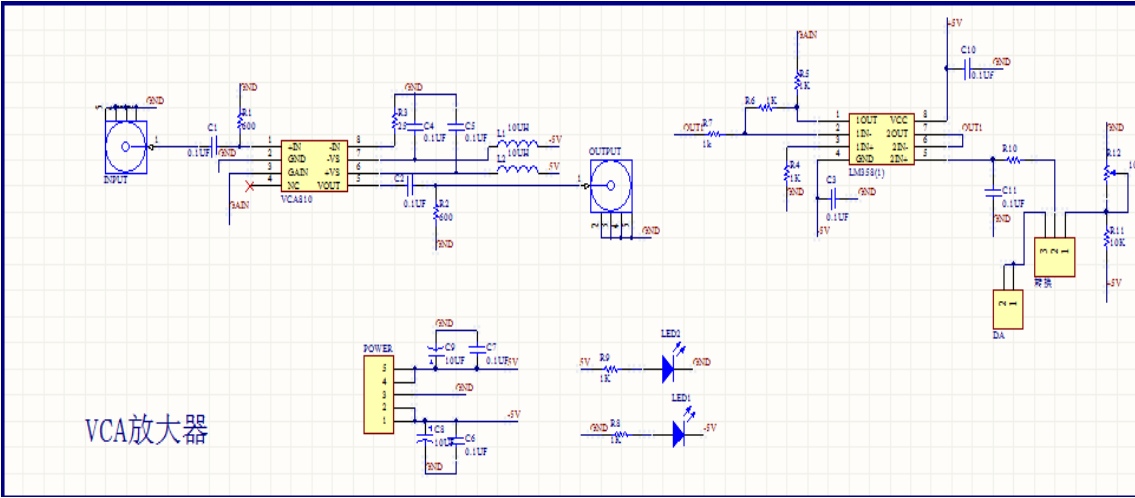


图 5 放大电路原理图

3.3 检波模块

电路设计采用三极管构成的跟随器形成三极管缓冲电路，对输入信号隔离，减少信号干扰，后级经过专用的检波芯片，以宽带的形式搭建检波电路进行测量峰值，最终波形接近放大器的理想幅频特性曲线。

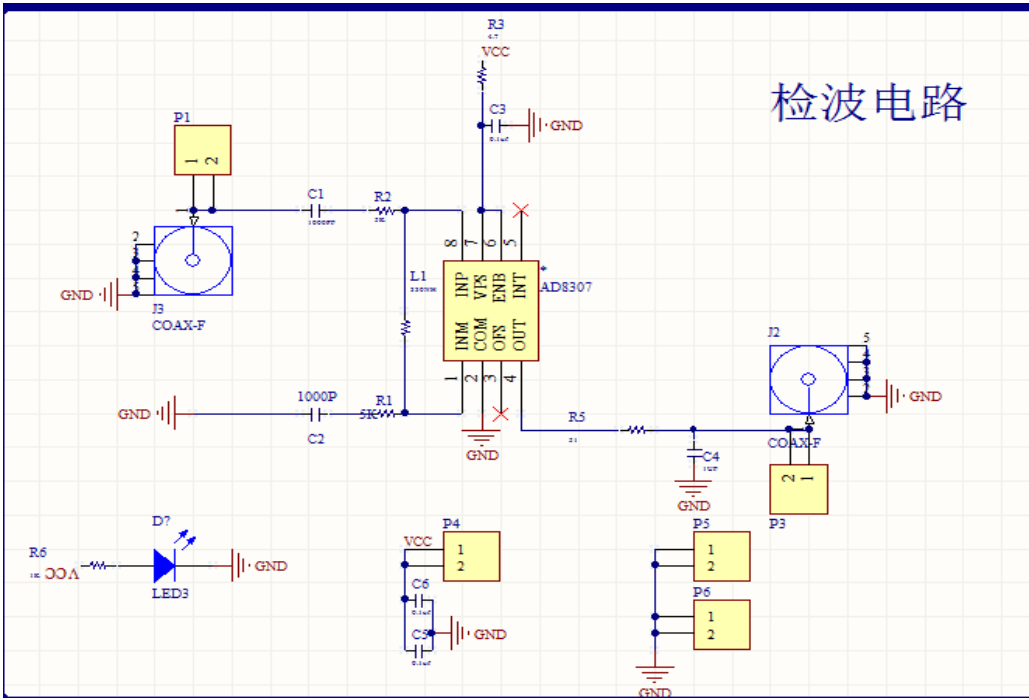


图 6 检波电路原理图

3.4 有线信道模块

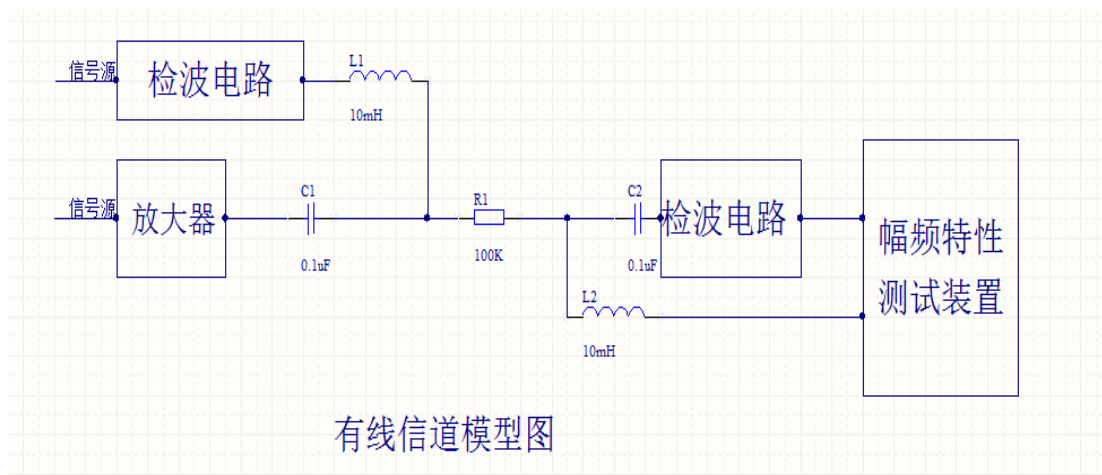


图 7 有线信道原理图

四、软件程序设计

4.1 单片机控制和显示设计

单片机采用 STC12 单片机作为控制核心，对 AD9854 信号源进行控制，同时提供键盘扫描和界面交互。软件流程图 8 所示：

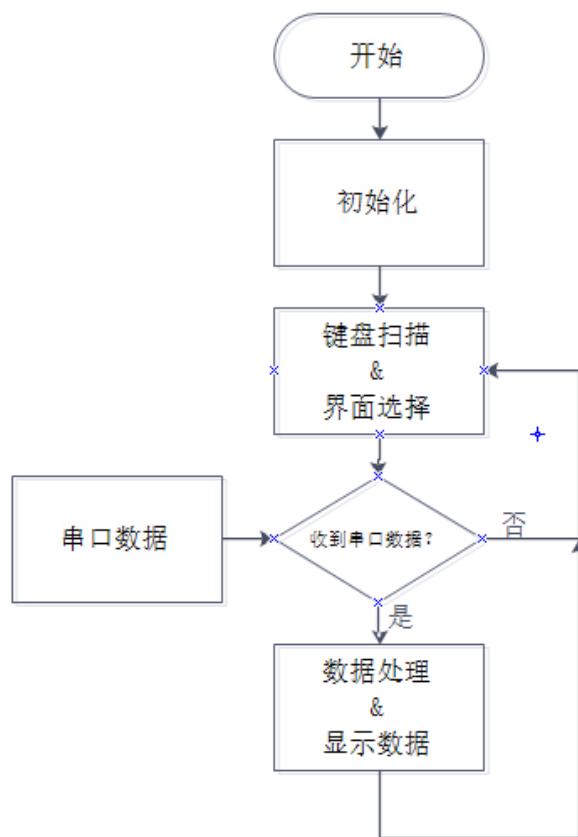


图 8 软件流程图

在本设计中，人机交互接口由 USART HMI 智能串口屏和按键组成。提供频率 1M 步进、幅值可调功能选择和对应的界面显示，文字显示简单，按键控制简单。在显示屏上，显示输出的频率值以及相应输出的幅度值。

4.2 FPGA 程序设计

本设计中，将信号源频率直接输入到 FPGA 中，通过 FPGA 进行频率测量，将频率值输出，作为示波器的 X 轴进行显示。同时在发挥部分中，将检波器得到的幅度信息通过 AD 输入到 FPGA 中，通过 WIFI 与笔记本电脑在路由器组成局域网中进行通信，从而在上位机上进行显示。软件功能如图 9 所示：

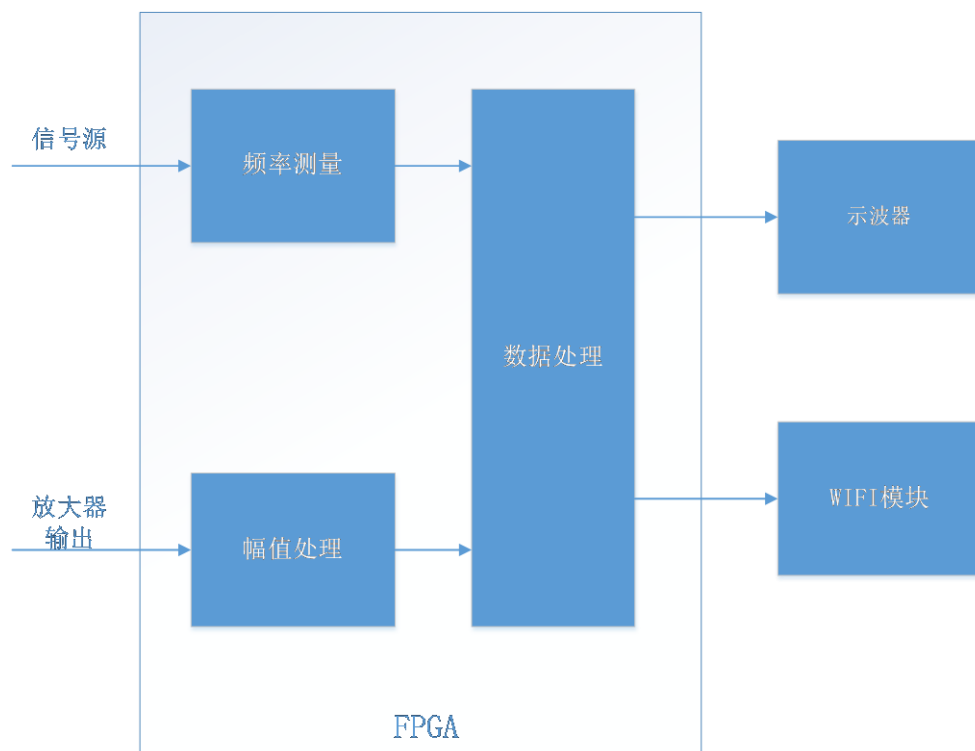


图 9 软件功能图示

4.3 局域网配置和上位机设计

本方案中，上位机组成几个部分，分别是转换电路 WiFi，路由器，上位机设计。具体组成见图 10 上位机组成图示

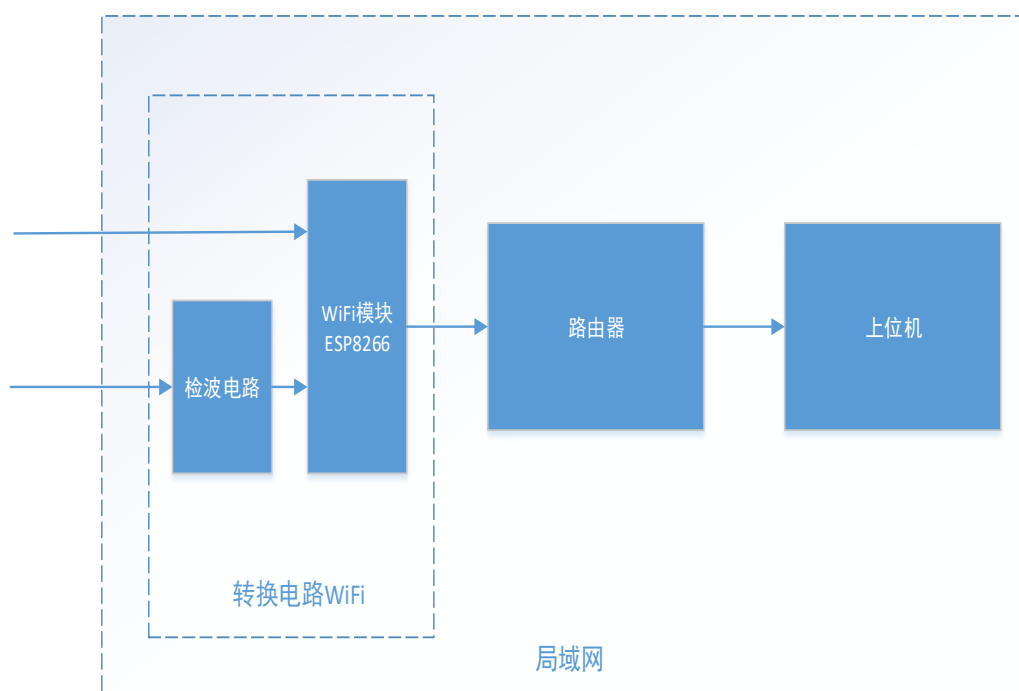


图 10 上位机组成图示

FPGA 作为下位机，将信号源频率信息、放大器的输出信号信息通过 WiFi 路由器自主搭建的局域网，发送到笔记本电脑上的上位机，由笔记本电脑完成放大器输出信号的幅频特性测试，并以曲线方式显示放大器输出信号的幅频特性。

WiFi 转换电路中，ESP8266 内嵌 32 位内核，透传模式下可固化指令。电脑通过串口对模块进行配置。最终具体操作界面如 图 11 WiFi 控制界面 所示：

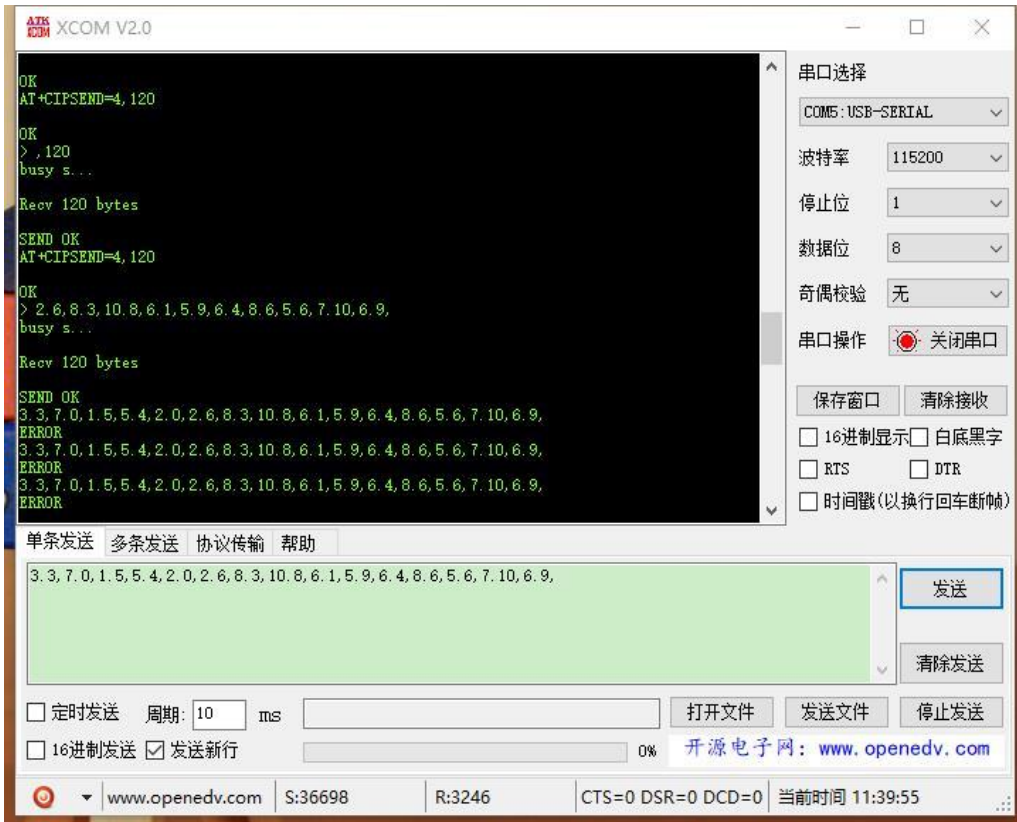


图 11 WiFi 控制界面

WiFi 通过 UDP 协议将信号源的频率信息、放大器的输出信息，发送到对应的笔记本电脑的 IP 地址，与电脑建立连接。

路由器通过官方的说明进行配置，路由器型号为 TP-LINK，为 150M 无线路由器。路由器 IP 为 192.168.1.1，登录 192.168.1.1，对路由器进行配置如图 12 所示：

版本信息	
当前软件版本:	5.6.21 Build 130417 Rel.73893n
当前硬件版本:	WR742N 5.0 00000000

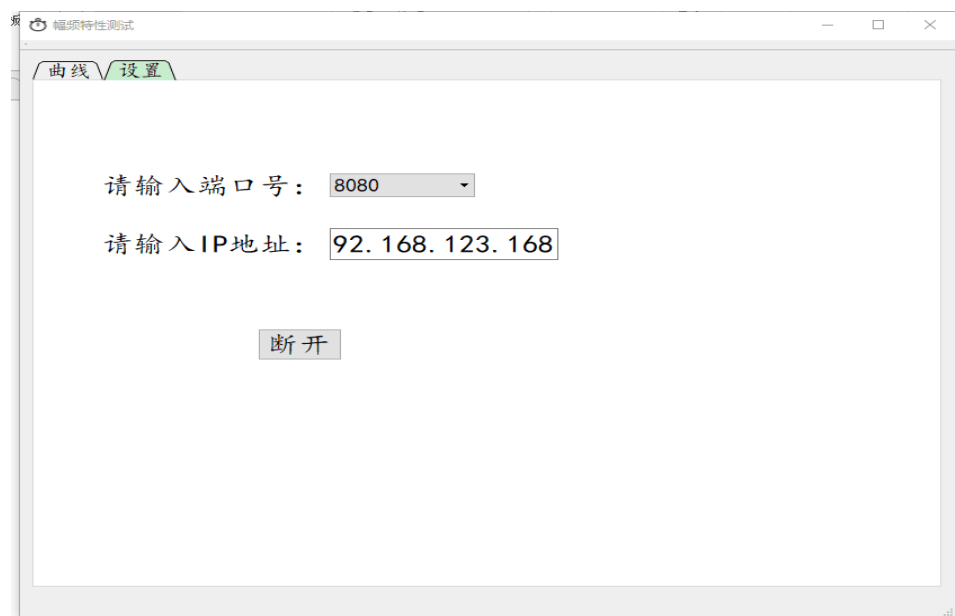
LAN口状态	
MAC地址:	A8-57-4E-76-81-7E
IP地址:	192.168.1.1
子网掩码:	255.255.255.0

无线状态	
无线功能:	启用
SSID号:	123
信道:	自动 (当前信道 6)
模式:	11bgn mixed
频段带宽:	自动
MAC地址:	A8-57-4E-76-81-7E
WDS状态:	未开启

WAN口状态	
MAC地址:	A8-57-4E-76-81-7F
IP地址:	211.86.111.22 动态IP
子网掩码:	255.255.255.192
网关:	211.86.111.62 释放
DNS服务器:	202.194.116.66 , 219.146.1.66

图 12 路由器配置

上位机配置界面



上位机配置界面截图显示了一个名为“幅频特性测试”的窗口。窗口顶部有“曲线”和“设置”两个标签，其中“设置”标签被选中。在“设置”标签下，有两个输入框：第一个是“请输入端口号:”，后面跟着一个下拉菜单，当前显示为“8080”；第二个是“请输入IP地址:”，后面跟着一个文本框，当前显示为“92.168.123.168”。在输入框下方，有一个“断开”按钮。

图 13 路由器配置

通过笔记本电脑连接路由器，与 WiFi 建立连接，解码接收信号源频率信息与放大器的输出信号信息，并进行运算得到放大器输出信号幅频特性曲线。

五、测试方案与结果

5.1 测试仪器

表 1：测试仪器列表

序号	名称	型号规格
1	数字存储示波器	GDS-1022
2	数字合成函数信号发生器	F20
3	数字多用表	TH1951
4	稳压源	SS
5	频率特性测试仪	NW1258

5.2 测试方案

1. 频率测试

使用 AD9854 DDS 信号源输出正弦波，频率 1MHz-40MHz，通过按键对于频率进行步进 1MHz 测试和自动扫描测试，用示波器进行波形观察。

2. 增益测试

使用 AD9854 DDS 信号源，提供一路扫频信号通过放大器，手动调节放大器的增益变化通过示波器进行观察。

3. 幅值测试

使用单片机输入相应的幅值数据，控制 AD9854 DDS 信号源产生对应波形，使用示波器测试，并与理想结果相对比。

5.3 测试结果

AD9854 信号源测试结果

产生 1M 波形

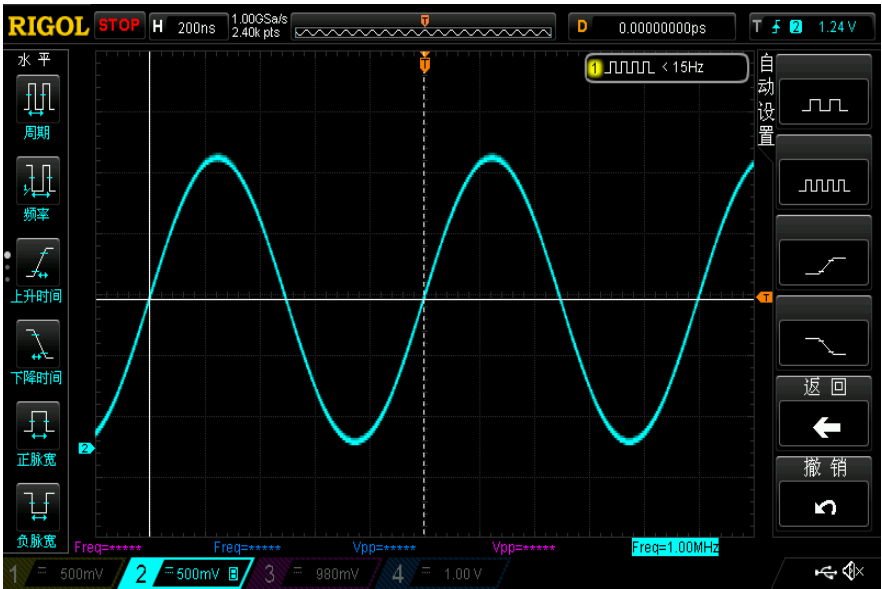


图 14 1M 波形效果

产生 10M 波形

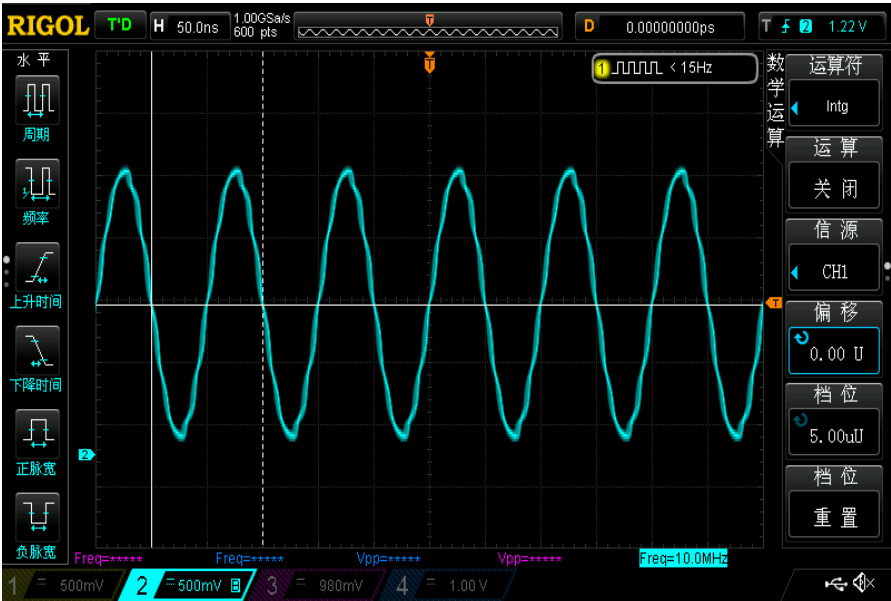


图 15 10M 波形数据

产生 40M 波形



图 16 40M 波形效果

测试数据见下列表格：（标准值为示波器显示参数）

表 2：信号源测试数据

频率(MHz)	1	5	10	20	40
输出(v)	1.02	0.98	1.01	1.02	0.98

表 3：放大器数据测试

频率(MHz)	1	5	10	20	22	30	35	40
输入(mv)	10	20	30	50	55	70	80	100
输出(v)	1.02	0.98	1.01	0.99	1	1.01	1.02	0.98

上位机测试结果：

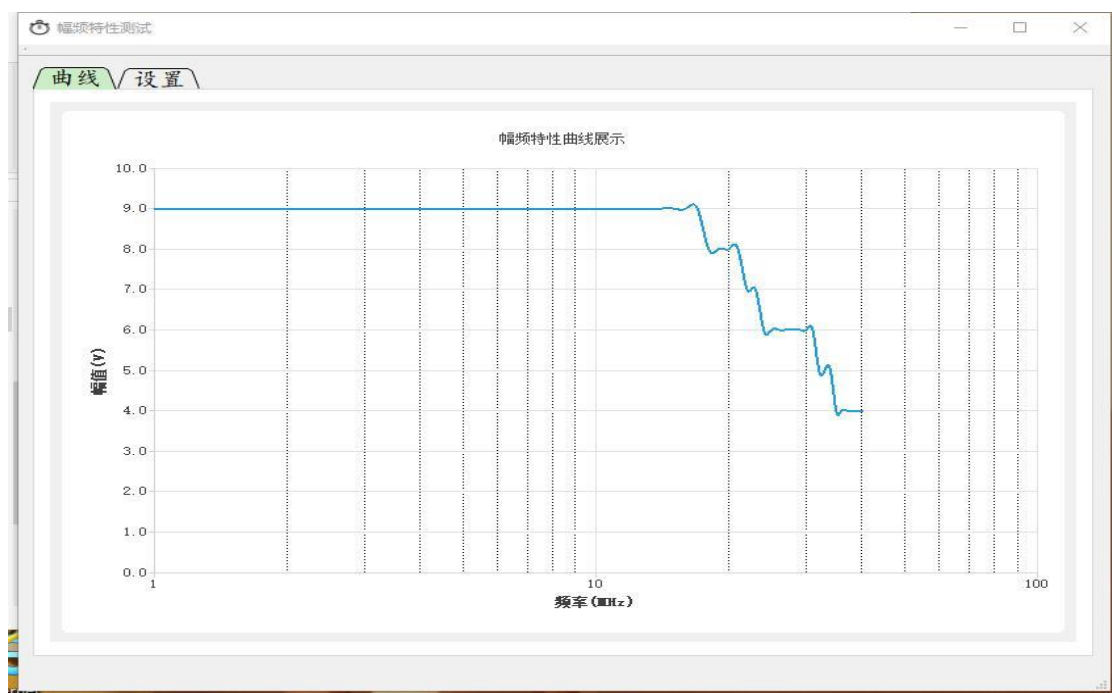


图 14 上位机幅频特性曲线

六、设计实现

1. 进一步提高测量精度，采用 FPGA 进行信号源数据以及放大器数据处理，可以准确对数据进行显示。
2. 模拟前端进一步改进，进一步提高系统的抗干扰能力，可通过优化布局，加屏蔽等方式。
3. 优化逻辑结构，尽量避免 FPGA 内部的竞争与冒险。

七、结论

本系统实现通过自制信号源，通过放大器后将信号源频率信息、放大器的输出信息，直接连接，或经过有线信道，或经过无线 WiFi 信道送至幅频特性测试装置进行测量，从而得到放大器输出信号的幅频特性曲线。整个系统效果稳定，存在一定误差，同时紧密结合“互联网+”，通过 WiFi 将测得信号发送到笔记本

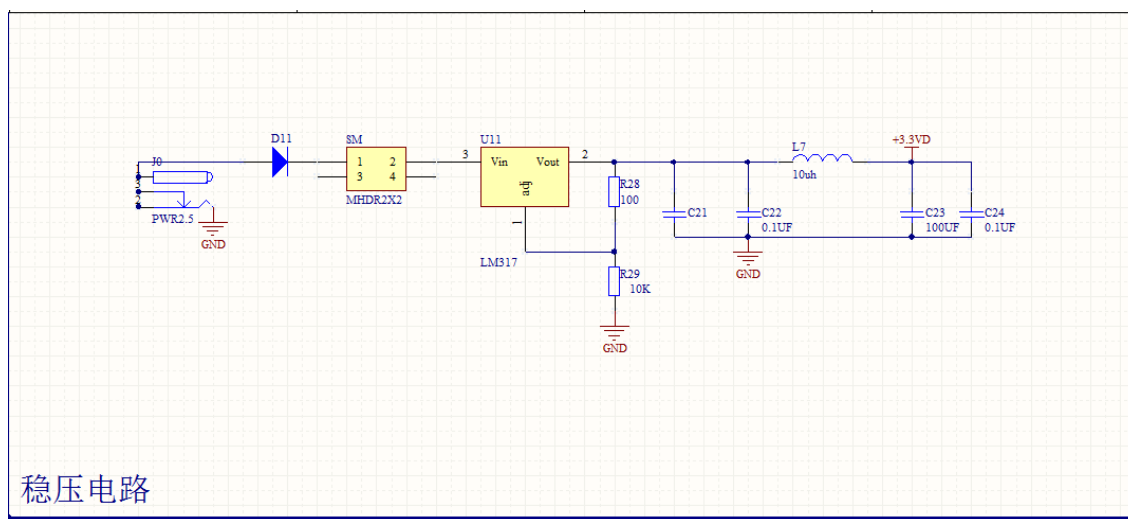
电脑进行运算，从而得到对应的曲线。

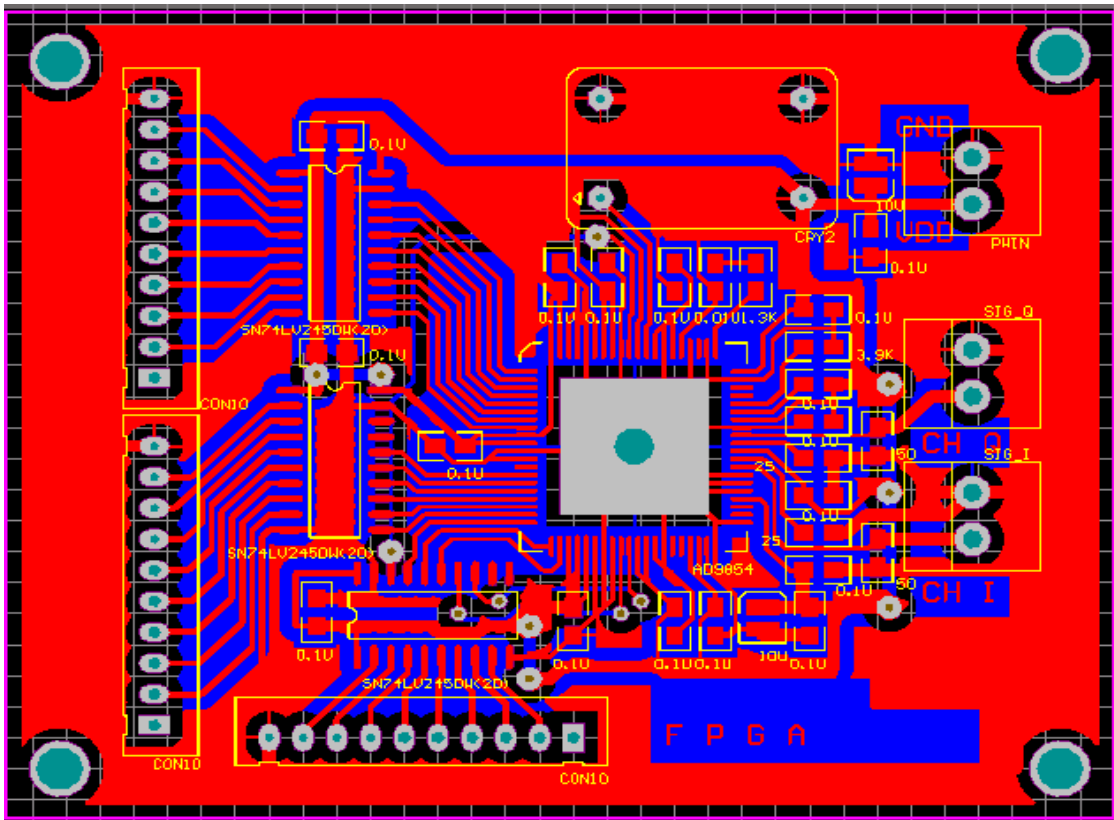
参考文献

- [1] 黄智伟. 全国大学生电子设计竞赛系统教程[M]. 电子工业出版社.2005 年.
- [2] 童诗白, 华成英. 模拟电子技术基础[M]. 高等教育出版社.2009 年.
- [3] 何宾, Altera 可编程逻辑器件技术详解[M].清华大学出版社, 2010 年.
- [4] 夏宇闻, 黄然等, Verilog SOPC 高级实验教程[M].北京航空航天大学出版社, 2011 年.

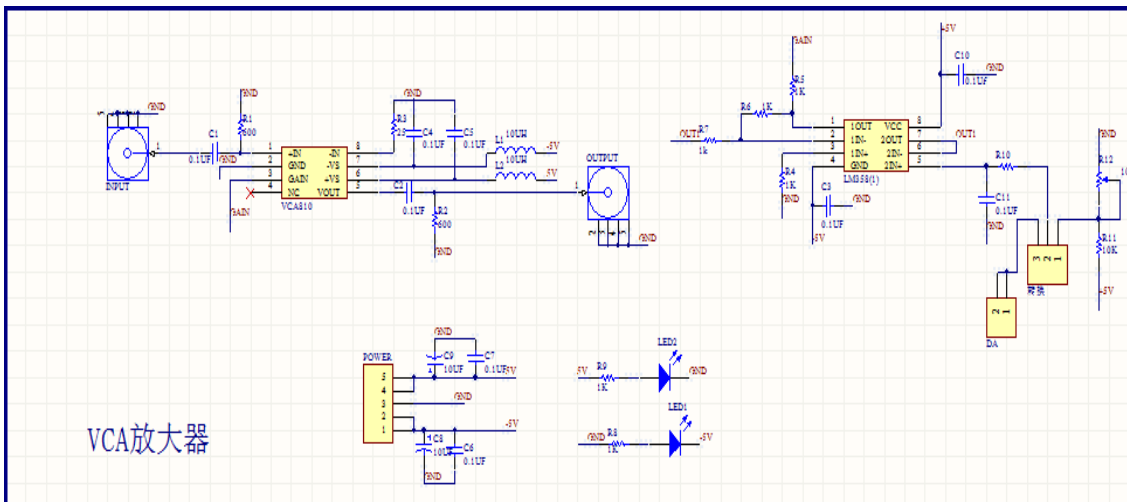
附录

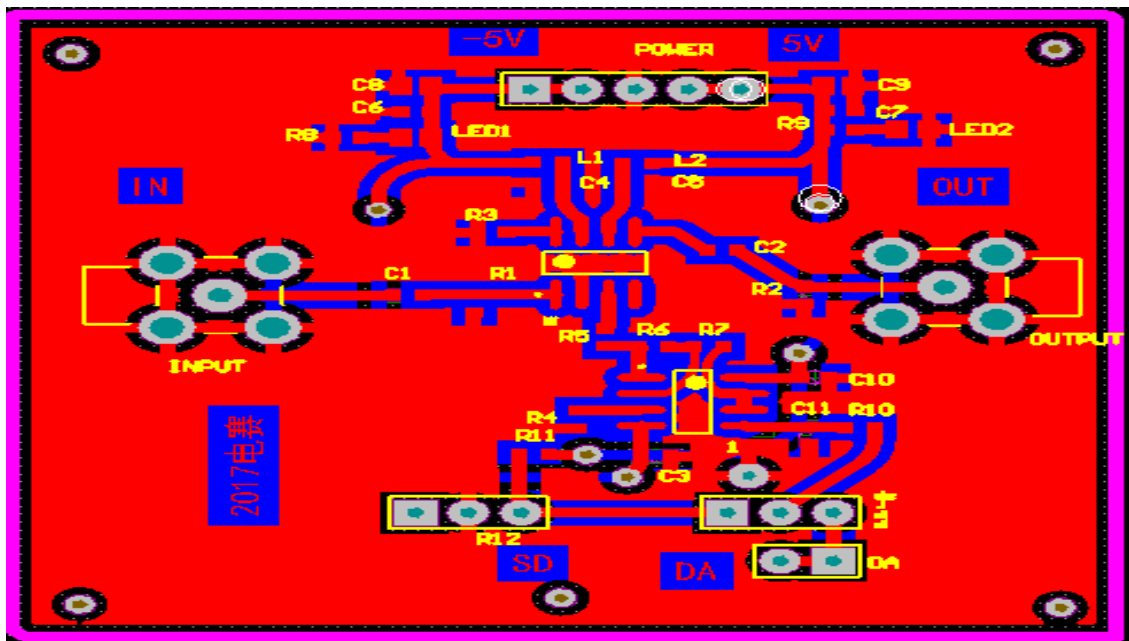
附录 1 AD9854 信号源电路



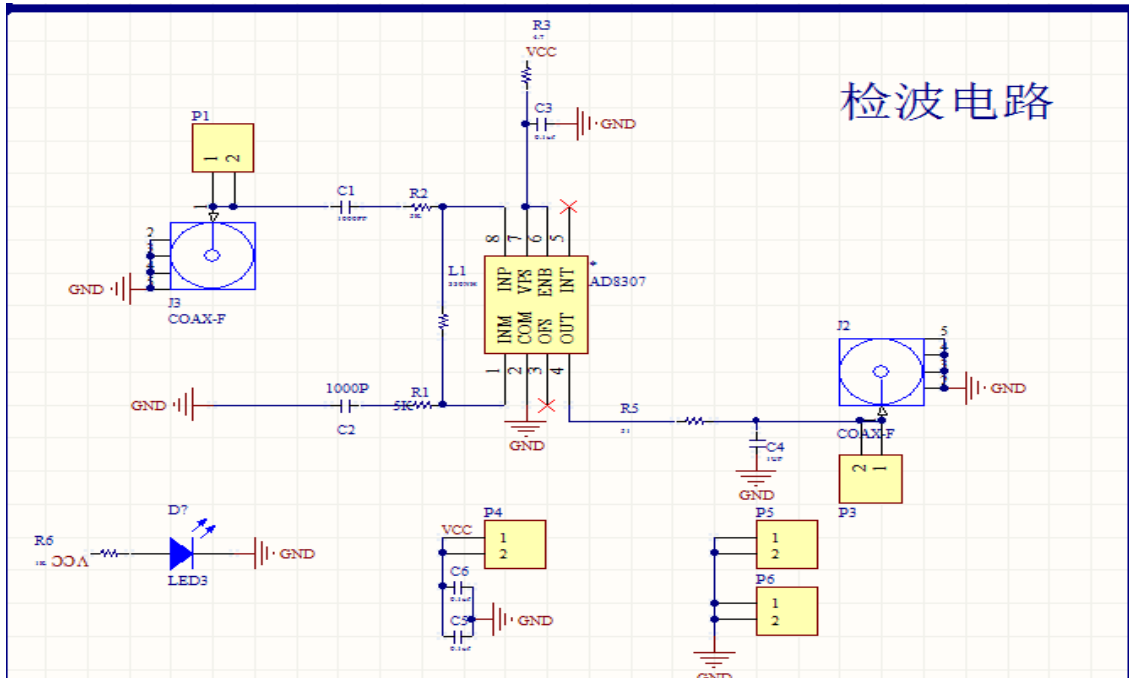


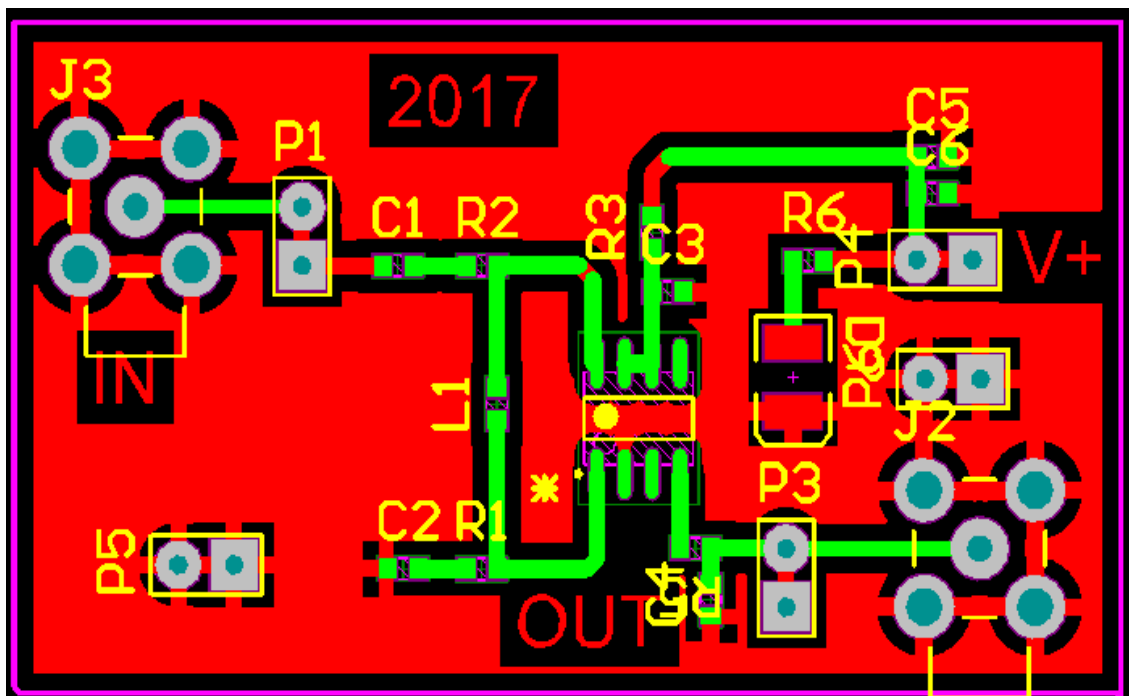
附录 2 VCA810 放大器电路





附录 3 AD8307 检波电路





附录 4 FPGA 下位机代码

```

/*-----
---

Date           :      2017-08-09
Description     :      Design for 2017 年电赛 .

-----
-*/

module pro_2017
(
    //global clock
    input        clk,           //system clock
    input        rst_n,         //sync reset

    //uart interface
    output       uart_tx,

    //sig_in  interface
    input        sig_in

```



```

);
//ceshi
//-----
//Funtion : ram数据
wire          wren;
wire          [7:0]  ram_data;
wire          [5:0]  addr_rd;
wire          [5:0]  addr_wd;
wire          [7:0]  q_ram;
ram_8_6      ram_inst
(
    .clock(clk),
    .data(ram_data),
    .rdaddress(addr_rd),
    .wraddress(addr_wd),
    .wren(wren),
    .q(q_ram)
);

//-----
//Funtion : ram_control
wire          [31:0]  pinlv;
wire          [7:0]   fuzhi;
ram_control ram_control_inst
(
    //global clock
    .  clk(clk),          //system clock
    .  rst_n(rst_n),      //sync reset

    //pinlv or fuzhi interface
    .  pinlv(pinlv),      //
    .  fuzhi(fuzhi),      //
    //ram_control

    .  ram_data(ram_data),
    .  addr_wd(addr_wd)
);

//-----
//Funtion : wifi_send
wire          [2:0]   wifi_state;

```

```

wifi_send  wifi_inst
(
    //global clock
    .   clk(clk),           //system clock
    .   rst_n(rst_n),       //sync reset

    //uart interface
    .   uart_tx(uart_tx),    //
    .   wifi_state(wifi_state),
    /nd_data interface
    .   addr_rd(addr_rd),    //读地址
    .   wren(wren),
    .   addr_wd(addr_wd),
    .   q_ram(q_ram)

);

//-----
//Funtion : 200M 倍频
wire      clk_200;

clk_200    clk_inst(
    .   refclk(clk),    // refclk.clk
    .   rst(!rst_n),    // reset.reset
    .   outclk_0(clk_200) // outclk0.clk
    // .   locked    // locked.export
);

//-----
//Funtion : 200M 倍频

cepin  cepin_inst
(
    //global clock
    .   clk(clk_200),    //system clock
    .   rst_n(rst_n),    //sync reset

    //pinlv interface
    .   sig_in(sig_in),

```

```

    //user interface

    .    pinlv(pinlv)

);

endmodule

```

附录 5 上位机 QT 界面代码

```

/*****
*****
** Meta object code from reading C++ file 'mainwindow.h'
**
** Created by: The Qt Meta Object Compiler version 67 (Qt 5.8.0)
**
** WARNING! All changes made in this file will be lost!
*****
*****/

#include "../mainwindow.h"
#include <QtCore/qbytearray.h>
#include <QtCore/qmetatype.h>
#if !defined(Q_MOC_OUTPUT_REVISION)
#error "The header file 'mainwindow.h' doesn't include <QObject>."
#elif Q_MOC_OUTPUT_REVISION != 67
#error "This file was generated using the moc from 5.8.0. It"
#error "cannot be used with the include files from this version of Qt."
#error "(The moc has changed too much.)"
#endif

QT_BEGIN_MOC_NAMESPACE
QT_WARNING_PUSH
QT_WARNING_DISABLE_DEPRECATED
struct qt_meta_stringdata_MainWindow_t {
    QByteArrayData data[4];
    char stringdata0[39];

```

```

};

#define QT_MOC_LITERAL(idx, ofs, len) \
    Q_STATIC_BYTE_ARRAY_DATA_HEADER_INITIALIZER_WITH_OFFSET(len, \
    qptrdiff(offsetof(qt_meta_stringdata_MainWindow_t, stringdata0) + ofs \
        - idx * sizeof(QByteArrayData)) \
    )
static const qt_meta_stringdata_MainWindow_t
qt_meta_stringdata_MainWindow = {
    {
QT_MOC_LITERAL(0, 0, 10), // "MainWindow"
QT_MOC_LITERAL(1, 11, 4), // "recv"
QT_MOC_LITERAL(2, 16, 0), // ""
QT_MOC_LITERAL(3, 17, 21) // "on_pushButton_clicked"

    },
    "MainWindow\0recv\0\0on_pushButton_clicked"
};

#undef QT_MOC_LITERAL

static const uint qt_meta_data_MainWindow[] = {

    // content:
    7,      // revision
    0,      // classname
    0,    0, // classinfo
    2,   14, // methods
    0,    0, // properties
    0,    0, // enums/sets
    0,    0, // constructors
    0,     // flags
    0,     // signalCount

    // slots: name, argc, parameters, tag, flags
    1,    0,   24,    2, 0x0a /* Public */,
    3,    0,   25,    2, 0x08 /* Private */,

    // slots: parameters
    QMetaType::Void,
    QMetaType::Void,

```

```

        0          // eod
};

void MainWindow::qt_static_metacall(QObject *_o, QMetaObject::Call _c,
int _id, void **_a)
{
    if (_c == QMetaObject::InvokeMetaMethod) {
        MainWindow *_t = static_cast<MainWindow *>(_o);
        Q_UNUSED(_t)
        switch (_id) {
            case 0: _t->recv(); break;
            case 1: _t->on_pushButton_clicked(); break;
            default: ;
        }
    }
    Q_UNUSED(_a);
}

const QMetaObject MainWindow::staticMetaObject = {
    { &QMainWindow::staticMetaObject,
qt_meta_stringdata_MainWindow.data,
    qt_meta_data_MainWindow, qt_static_metacall, Q_NULLPTR, Q_NULLPTR}
};

const QMetaObject *MainWindow::metaObject() const
{
    return QObject::d_ptr->metaObject ?
QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
}

void *MainWindow::qt_metacast(const char *_cname)
{
    if (!_cname) return Q_NULLPTR;
    if (!strcmp(_cname, qt_meta_stringdata_MainWindow.stringdata0))
        return static_cast<void*>(const_cast< MainWindow*>(this));
    return QMainWindow::qt_metacast(_cname);
}

int MainWindow::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
{

```

```
_id = QMainWindow::qt_metacall(_c, _id, _a);
if (_id < 0)
    return _id;
if (_c == QMetaObject::InvokeMetaMethod) {
    if (_id < 2)
        qt_static_metacall(this, _c, _id, _a);
    _id -= 2;
} else if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
    if (_id < 2)
        *reinterpret_cast<int*>(_a[0]) = -1;
    _id -= 2;
}
return _id;
}
QT_WARNING_POP
QT_END_MOC_NAMESPACE
```