1　千兆以太网传输 ACM1030 数据采集

工程源码	|-ac6103开发板 |-- ac6103_acm1030_rgmii
相关视频课程	

章节导读

本节内容将介绍基于 ACM1030 模块利用网口进行数据采集的相关内容。FPGA 采用 RGMII 接口与以太网 PHY 芯片 RTL8211 通信，接收以太网数据包， 并提取出以太网数据包中 UDP 协议报文的数据内容，然后将数据转化成控制命令，从而实现对 ACM1030 模块的采样频率、数据采样个数以及采样通道的合理配置，采集完成后的数据经 FIFO 缓存后，通过网口以 UDP 协议传输到电脑。用户可以在电脑上通过网口调试工具进行指令的下发，并以文件的形式保存接收到的数据，然后使用 MATLAB 软件进行进一步的数据处理分析。

1.1 系统整体设计

通过电脑上的网络调试助手，将命令帧进行发送，然后通过 AC6103 开发板上的以太网芯片接收，随后将接收到的数据转换成命令， 从而实现对 ACM1030 模块采样频率、数据采样个数以及采样通道的配置。配置完成之后，ACM1030 模块开始采集数据，将采集的数据存储至 FIFO 中， 在配置寄存器的过程中要考虑到 FIFO 的写深度， 一次采样所能采集的数据应该小于或等于 FIFO 的写深度。最后数据通过网口传输至电脑，电脑端将采集到的数据通过 MATLAB 进行进一步的分析，系统的整体设计框图如下图 1-1 所示。

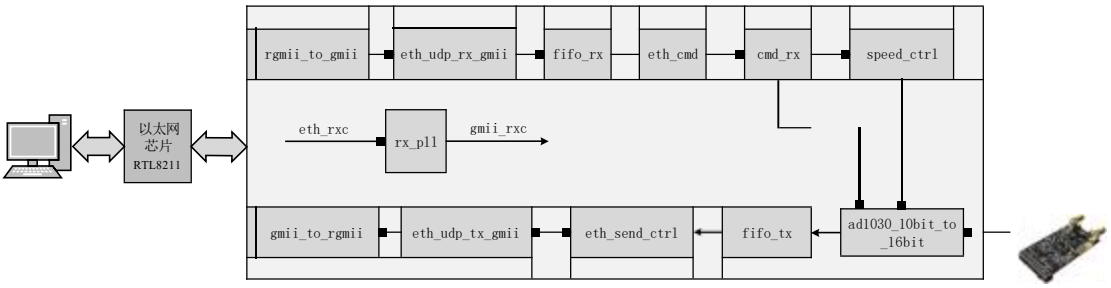

图 1-1 千兆以太网传输 ACM1030 数据采集整体设计框图

对于上图中各个模块的功能介绍如下：

1. gmii_rxc 模块：锁相环模块， 将 rgmii 接口时钟信号 eth_rxc 偏移 135°得到 gmii_rxc 时钟信号。

2. rgmii_to_gmii 模块：以太网接收 rgmii 转 gmii，将 rgmii 接口信号转换成 gmii 接口。

3. eth_udp_rx_gmii 模块： GMII 以太网接收模块，接收用户在电脑上通过网络助手或者上位机发送的包含指令数据的数据帧。

4. fifo_rx 模块： FIFO IP 核，以太网接收模块工作时钟 125M，ACM1030 数据采集模块的工作时钟 50M，两者数据速率不匹配，使用该 IP 核解决采集过程中会出现的跨时钟域数据交互问题。

5. eth_cmd 模块：接收转命令模块，对接收到的以太网数据进行分析，提取出每个控制命令帧。

6. cmd_rx 模块：指令转控制模块，将从接收转命令模块接收到的数据转换为相应的控制数据并分别输出到对应的模块。

7. speed_ctrl 模块：采样速率控制模块，控制 ACM1030 的采样速率。

8. ad1030_10bit_to_16bit 模块： 将 ACM1030 采集到的 10 位数据转换成 16 位的有符号数据，这样做的目的是为了方便计算机进行数据存储。

9. fifo_tx 模块： FIFO IP 核，存储从 ad1030_10bit_to_16bit 模块输出的 16 位数据，数据经缓存后又由以太网发送模块读取。

10. eth_send_ctrl 模块：网口发送控制模块， 该模块主要控制网口发送模块的使能控制信号以及对以太网数据帧数据长度的控制。

11. eth_udp_tx_gmii 模块：网口发送模块，将采集到的数据以以太网帧的形式进行发送。

12. gmii_to_rgmii 模块：以太网发送 gmii 转 rgmii，将 gmii 接口信号转换成 rgmii 接口。

本章将讲解的模块包括eth_cmd 模块、 cmd_rx 模块、 speed_ctrl 模块、ad1030_10bit_to_16bit 模块和 eth_send_ctrl 模块，以太网部分的模块请读者查看以太网相关章节的内容。

1.2 ACM1030 模块简介

ACM1030 模块是基于国产知名模拟器件设计和制造商思瑞浦（3PEAK）公司的 10 位 50M 采样速率高速 ADC 芯片 3PA1030进行设计的，该模块如下图 1-2 所示。 ACM1030 模块配合前端模拟信号调理电路，实现了±5V 电压范围内信

号的高速采样。该模块共使用 2 路完全相同的 AD 采样和信号调理电路，构成了双通道高速 AD 采样电路。两路 ADC 电路完全独立，结构和元器件参数相同，确保了两个通道有较高的一致性。本模块与 FPGA 连接采用并行接口，每路 ADC 包括 10 位数据信号（ADC_DATA），1 位时钟信号（ADC_CLK），1 位超量程指示信号（ADC_OVR），该模块接口图如下图 1-3 所示。

图 1-2 ACM1030 模块图

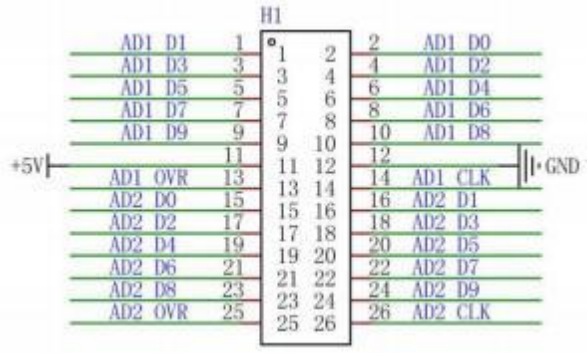

图 1-3 ACM1030 模块接口图

使用该模块时，仅需 FPGA 为每路ADC 提供一路时钟信号， ADC 则会每个时钟周期输出一个 10 位的采样结果。 当 3PA1030 模拟输入端接-5V 至+5V 之间变化的正弦波电压信号时，其转换后的数据也是成正弦波波形变化， 转换波形如下图 1-4 所示，从图中可以看出 3PA1030 采集到的数据是无符号数据。

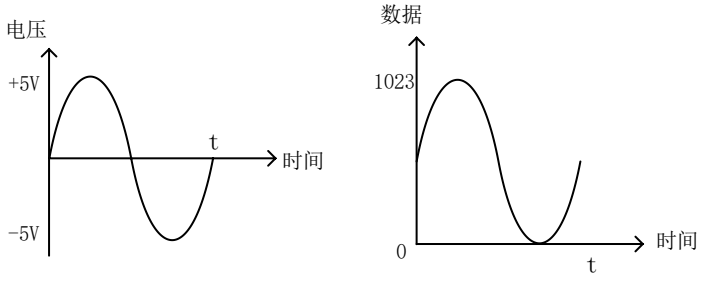

图 1-4 3PA1030 正弦波模拟电压值（左）、数据（右）

本模块采样率上限为 50Msps，采样率就等于 FPGA 提供给 ADC 的时钟频率。如需使用低于时钟频率的采样率，可以依旧给 ADC 提供 50MHz 的时钟信号，但在 FPGA 内部， 对 50Msps 的采样结果数据进行抽取重采样的方法实现。比如期望以 1Msps 的采样速率采样，则只需要每间隔 50 个采样数据取一个结果存储或使用，其他 49 个数据直接舍弃，这样就能实现 1MSPS 的采样率了。十分不建议采用直接对提供给 ADC 芯片的时钟信号降频以实现降低采样率的效果的方法，因为时钟太低，会影响 ADC 芯片内部采样保持电路的工作情况，导致采样误差偏大。

本模块可用于小梅哥全系列 FPGA、SOC、Zynq 开发板， 包括国产开发板和各核心板的评估底板。 AC620、AC6103、ACX720、ACZ702、AC609、智多晶 FPGA 开发板（AC208-SA5Z）、AC608 评估底板、 AC601 评估底板、 AC675 评估底板。

1.3 模块设计

下面给将对本次实验需要设计的模块进行介绍。

1.3.1 接收转命令模块

接收转命令模块 eth_cmd，将以太网传输过来的指令数据帧进行拆解，得到需要的指令数据传送给别的模块进行处理，该模块的结构框图如下图 1-5 所示。

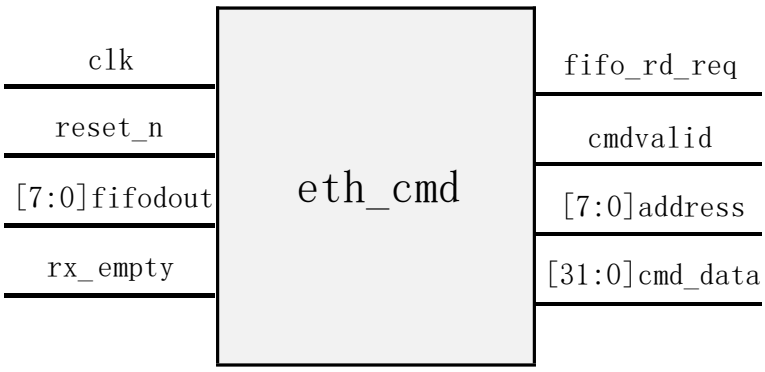

图 1-5 接收转命令模块框图

模块信号说明如下表 1-1 所示。

表 1-1 接收转命令模块信号说明表

信号名称	I/O	信号意义
clk	I	模块工作时钟
reset_n	I	模块复位信号，低电平复位
fifodout[7:0]	I	从 FIFO 中读出的 8 位数据

rx_empty	I	FIFO 为空标志信号
fifo_rd_req	O	FIFO 的读请求信号
cmdvalid	O	输出命令有效标志信号
address[7:0]	O	配置 AD 模块的寄存器地址信号
cmd_data[31:0]	O	写入到寄存器中的数据

网口一次发送的命令数据内容为 32 个字节，为了实现通过网口修改这些寄存器的值，需要对发送一次的数据进行拆解才能实现，对于设计的数据帧， 一帧数据一共 8 个字节， 包含帧头、帧尾、地址段、数据段。帧格式如下表 1-2 所示：

表 1-2 帧格式说明表

数据	D0	D1	D2	D3	D4	D5	D6	D7
功能	帧头 0	帧头 1	地址 address	data[31:24]	data[23：16]	data[15:8]	data[7:0]	帧尾
值	0x55	0xA5	XX	XX	XX	XX	XX	0xF0

从上表中可以看出，每帧数据一共 8 个字节， 分别用 D0~D7 表示，其中，D0 和 D1 两个数据作为帧头，其值固定为 0x55、0xA5，D7 作为帧尾，其值固定为 0xF0。帧头和帧尾的作用是为了准确识别数据帧，确保接收的数据是我们需要分析的。D2 代表的是要操作的寄存器地址，D3 为要写入寄存器的数据的 24~31 位， D4 为要写入寄存器的数据的 16~24 位， D5 为要写入寄存器的数据的 8~15 位， D6 为要写入寄存器的数据的 0~7 位。

该模块的作用就是将网口接收到的数据拆解成上述帧格式，将 D2 作为地址 address 输出，指定修改哪个寄存器， D3~D6 共 32 位作为数据 data 输出，控制 ACM1030 模块进行相应的配置。下面将对模块中的部分代码进行说明：

首先，产生 FIFO 的读请求信号。当检测到 FIFO 非空的时候，产生 FIFO 读请求信号，代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
  fifo_rd_req <= 1'b0;
else if(!rx_empty)
  fifo_rd_req <= 1'b1;
else
  fifo_rd_req <= 1'b0;
```

然后是得到帧命令数据，每产生一次读请求，将会从 FIFO 中读取一个 8 位的数据，连续存储 8 个字节的数据就得到一帧命令数据。代码如下所示：

```
always@(posedge clk)
if(fifo_rd_req)begin
  data_0[7] <= #1 fifodout;
  data_0[6] <= #1 data_0[7];
  data_0[5] <= #1 data_0[6];
```

```
  data_0[4] <= #1 data_0[5];
  data_0[3] <= #1 data_0[4];
  data_0[2] <= #1 data_0[3];
  data_0[1] <= #1 data_0[2];
  data_0[0] <= #1 data_0[1];
end
```

最后是判断得到的帧命令数据是否正确，当数据符合 D0 为 8'h55，D1 为 8'hA5，D7 为 8'hF0，则代表该数据格式正确，会生成一个指令正确信号 cmdvalid 输出到指令转控制模块，并将数据进行输出，代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)begin
  address <= 0;
  cmd_data <= 32'd0;
  cmdvalid <= 1'b0;
end
else if(fifo_rx_done)begin
if((data_0[0]==8'h55)&&(data_0[1]==8'hA5)&&(data_0[7]==8'hF0))begin
        cmd_data[7:0] <= #1 data_0[6];
        cmd_data[15:8] <= #1 data_0[5];
        cmd_data[23:16] <= #1 data_0[4];
        cmd_data[31:24] <= #1 data_0[3];
        address <= #1 data_0[2];
        cmdvalid <= #1 1;
    end
else
    cmdvalid <= #1 0;
end
```

1.3.2 指令转控制模块

指令转控制模块 cmd_rx 将从接收转命令模块接收到的数据转换为相应的控制数据，首先将对寄存器进行说明，其功能和地址分别如下表 1-3 所示：

表 1-3 寄存器说明表

名称	地址	位宽	功能简介
RestartReq	0	1	重新开始采集请求寄存器， 向该寄存器写入任意值即可启动新一轮的采样存储传输
ChannelSel	1	2	通道设置寄存器，共 2 位。ACM1030 模块提供了 ADC1、ADC2 两个通道进行数据采集。
DataNum	2	32	数据个数寄存器。如果采样 512 个数据，应该向寄存器中写入 0100。
ADC_Speed_Set	3	32	ADC 采样速率设置寄存器。如果设置为 0，采样和时钟保持一致 50M 时钟就是 50M 的采样速率，设置计数值后就可以改变采样频率，设置为 1 就是 25M 。如果设置为 27 0F，换算成十进制是 9999，采样速率设置是 5k，计数值和采样频率之间的关系： 设置计

			数值= Fclk/Fs－1，Fs 是期望的采样率， Fclk 是系统时钟 50M。

指令转控制模块的结构框图如下图 1-6 所示。

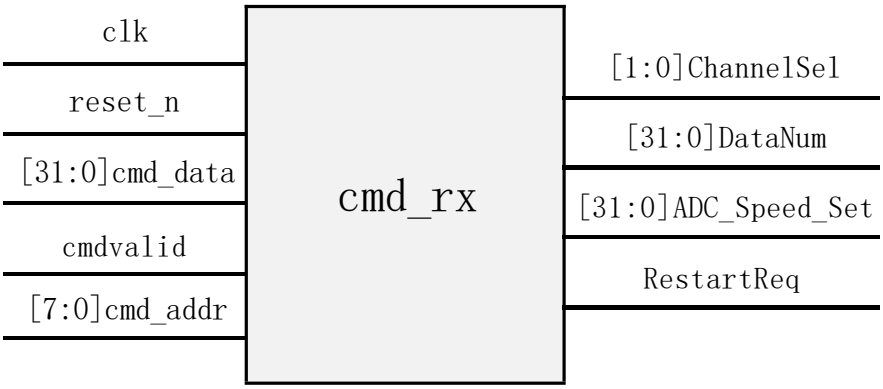

图 1-6 指令转控制模块结构框图

模块信号说明如下表 1-4 所示。

表 1-4 指令转控制模块信号说明表

信号名称	I/O	信号意义
clk	I	模块时钟信号
reset_n	I	模块复位信号，低电平复位
cmd_data[31:0]	I	写入到寄存器中的值
cmdvalid	I	命令有效标志信号
cmd_addr[7:0]	I	寄存器地址信号
ChannelSel[1:0]	O	通道设置寄存器
DataNum[31:0]	O	数据个数寄存器
ADC_Speed_Set[31:0]	O	ADC 采样速率控制寄存器
RestartReq	O	重新开始采集请求信号

根据表 1-3 中的内容， 地址 cmd_addr 为 0 时，产生 RestartReq；cmd_addr 为 1 时，得到通道设置数据 cmd_data[1:0]；cmd_addr 为 2 时，得到需要采样的数量 cmd_data[31:0]；cmd_addr 为 3 时，得到设置的采样速率的值 cmd_data[31:0]，代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)begin
    ChannelSel <= 2'b00;
    DataNum <= 32'd0;
    ADC_Speed_Set <= 32'd0;
    RestartReq <= 1'b0;
end
else if(cmdvalid)begin
  case(cmd_addr)
      0: RestartReq <= 1'b1;
      1: ChannelSel <= cmd_data[1:0];
      2: DataNum <= cmd_data[31:0];
```

```
      3: ADC_Speed_Set <= cmd_data[31:0];
  default:;
  endcase
end
else
      RestartReq <= 1'b0;
```

1.3.3 采样速率控制模块

采样速率控制（speed_ctrl）模块用来控制ACM1030 的采样速率，该模块的结构框图如下图 1-7 所示。

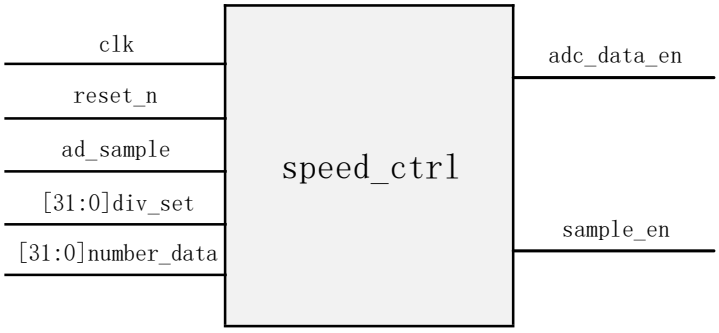

图 1-7 采样速率控制模块

对该模块的信号说明如下表 1-5 所示。

表 1-5 采样速率控制模块信号说明表

信号名称	I/O	信号意义
clk	I	模块时钟信号
reset_n	I	模块复位信号，低电平复位
ad_sample	I	输入的启动采样标志信号
div_set[31:0]	I	采样频率数据控制信号，div_set= Fclk/Fs – 1，Fs 是期望的采样率， Fclk 是系统时钟 50M
number_data [31:0]	I	需要采样的数据个数
adc_data_en	O	ADC 采样结果存储使能信号
sample_en	O	采样使能标志信号

ACM1030 模块的最大采样速率为 50M，如需使用低于时钟频率的采样率，可以依旧给 ADC 提供 50MHz 的时钟信号，但在 FPGA 内部，对 50Msps 的采样结果数据进行抽取重采样的方法实现。比如期望以 1Msps 的采样速率采样，则只需要每间隔 50 个采样数据取一个结果存储或使用，其他49 个数据直接舍弃，这样就能实现 1MSPS 的采样率了。 该模块可以通过状态机的方式实现所需功能，每个状态的代码设计如下所示：

状态0，当接收到启动采样信号时，产生采样使能标志信号，然后跳转到状态 1，代码如下所示：

```
if(ad_sample)begin
    sample_en <= 1'b1;
    state <= 1;
end
else begin
    sample_en <= 1'b0;
    state <= 1'b0;
end
```

状态 1，产生 ADC 采样数据使能信号 adc_data_en，控制 ADC 采样数据输出，代码如下所示：

```
begin
    adc_data_en <= 1;
    state <= 2;
end
```

状态 2，当产生 adc_data_en 信号之后，对采样数据进行计数。当采样的数据 data_cnt 小于需要采样的数据个数number_data 并且 div_set > 0 时（采样速率不使用 50M），跳转到状态 3；当采样数据 data_cnt 大于 number_data 并且 number_data 大于 0 时（使用 50M 采样率），跳转到状态 0，并且令 data_cnt、adc_data_en 等信号为 0，此时代表数据采样完成。代码如下所示：

```
if(adc_data_en)begin
    data_cnt <= data_cnt + 1'b1;
    if((data_cnt <= (number_data-1'b1)) && (div_set > 0))begin
        state <= 3;
        adc_data_en <= 1'b0;
    end
    if (data_cnt >= (number_data-1'b1) && number_data>0)begin
        sample_en <= 1'b0;
        adc_data_en <= 1'b0;
        data_cnt <= 32'd0;
        state <= 0;
    end
end
```

状态 3，针对采样速率不是 50M 的情况。进入该状态之后，计数器 div_cnt 开始计数，当 div_cnt 计数到 div_set-1 时，令 adc_data_en=1，使 ADC 模块采样输出，然后跳转到状态 2 进行判断。这样就实现每隔 div_set 个采样数据取一个结果存储或使用，从而达到对 ADC 采样频率的控制。 代码如下所示：

```
begin
    div_cnt <= div_cnt + 1'b1;
    if(div_cnt == div_set-1)begin
        div_cnt <= 0;
        adc_data_en <= 1;
        state <= 2;
```

```
    end
    else begin
        adc_data_en <= 1'b0;
        state <= 3;
    end
end
```

1.3.4 数据位扩展模块

数据采集模块 ACM1030 采集到的 10bit 数据不便于计算机存储，因为计算机对数据进行分析、存储的时候都以 8 位或 16 位数据作为统一的存储标准，所以我们需要通过数据位扩展模块（ad1030_10bit_to_16bit）将 10bit 数据转换成 16bit 数据进行存储。 该模块的结构框图如下图 1-8 所示。

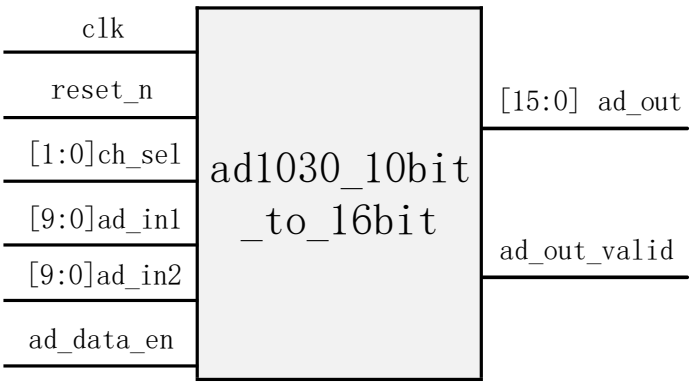

图 1-8 数据位扩展模块结构框图

对该模块的信号说明如下表 1-6 所示。

表 1-6 数据位扩展模块信号说明表

信号名称	I/O	信号意义
clk	I	模块时钟信号
reset_n	I	模块复位信号，低电平复位
ch_sel[1:0]	I	通道设置信号
ad_in1[9:0]	I	ACM1030 通道 1 的 10 位数据输入信号
ad_in2[9:0]	I	ACM1030 通道 2 的 10 位数据输入信号
ad_data_en	I	控制 ADC 采样数据使能信号
ad_out[15:0]	O	16 位数据输出信号
ad_out_valid	O	输出数据有效信号

下面将编写模块实现代码。

首先， 产生输出数据有效信号， 将 ad_data_en 信号给到 ad_out_valid，代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
    ad_out_valid <= 1'b0;
```

```
else
    ad_out_valid = ad_data_en;
```

然后将 ADC 采集到的无符号数据转换成有符号数据。如果采集的波形为+5V~-5V 的正弦波， ADC 模块最终输出的数据就为 1023~0 的正弦波， 但是上位机在分析数据的时候需要数据是有符号的，在这里我们进行的操作就是将 ADC 采集得到的数据加上512，也就是将最高位取反，最后进行分析时将最高位作为符号位。举个例子，如果 ADC 采集到的数据分别为 0、511、1023，将这些数据分别加上 512 之后得到的二进制值分别为 1000000000（-0）、1111111111（-511）、0111111111（+511），这样将最高位作为符号位，采样的数据就变成了有符号的数据，从而可以提供给我们的上位机进行数据分析。代码如下所示：

```
assign s_ad_in1 = ad_in1 + 10'd512;
assign s_ad_in2 = ad_in2 + 10'd512;
```

最后模块根据选择通道（ch_sel） 的不同，输出对应通道 的数据 。当 ch_sel= 2'b01（0x01），输出通道 1 的数据； 当 ch_sel= 2'b10（0x02），输出通道 2 的数据。 ADC 采集的数据是 10 位，这里通过补 0 的方式，实现 16 位的数据输出。代码如下所示：

```
always @(posedge clk or negedge reset_n)
if(!reset_n)
    ad_out <= 16'd0;
else if(ad_data_en && ch_sel == 2'b01)
    ad_out<={4'd0,s_ad_in1,2'd0};//这样补 0 为了适应上位机
else if(ad_data_en && ch_sel == 2'b10)
    ad_out<={4'd0,s_ad_in2,2'd0};//
else if(ad_data_en && ch_sel == 2'b00)
    ad_out<={4'd0,adc_test_data,2'd0};//
else
    ad_out <= ad_out;
```

1.3.5 FIFO 数据存储模块

FIFO 模块需要接收从数据位扩展模块输出的 16 位数据，数据经缓存后由以太网发送模块读取。 ADC 采集控制模块的工作时钟为 50M，以太网发送模块的工作时钟 125M，两者数据速率不匹配，使用 FIFO IP 核进行数据存储可以解决采集过程中会出现的跨时钟域数据交互问题。

在 Quartus 软件中依次点击 Tools-> MegaWigard Plug In Manager，然后在搜索栏中输入 FIFO， 在 下面搜索的结果中找到 FIFO Generator并双击，操作如下图 1-9 所示。

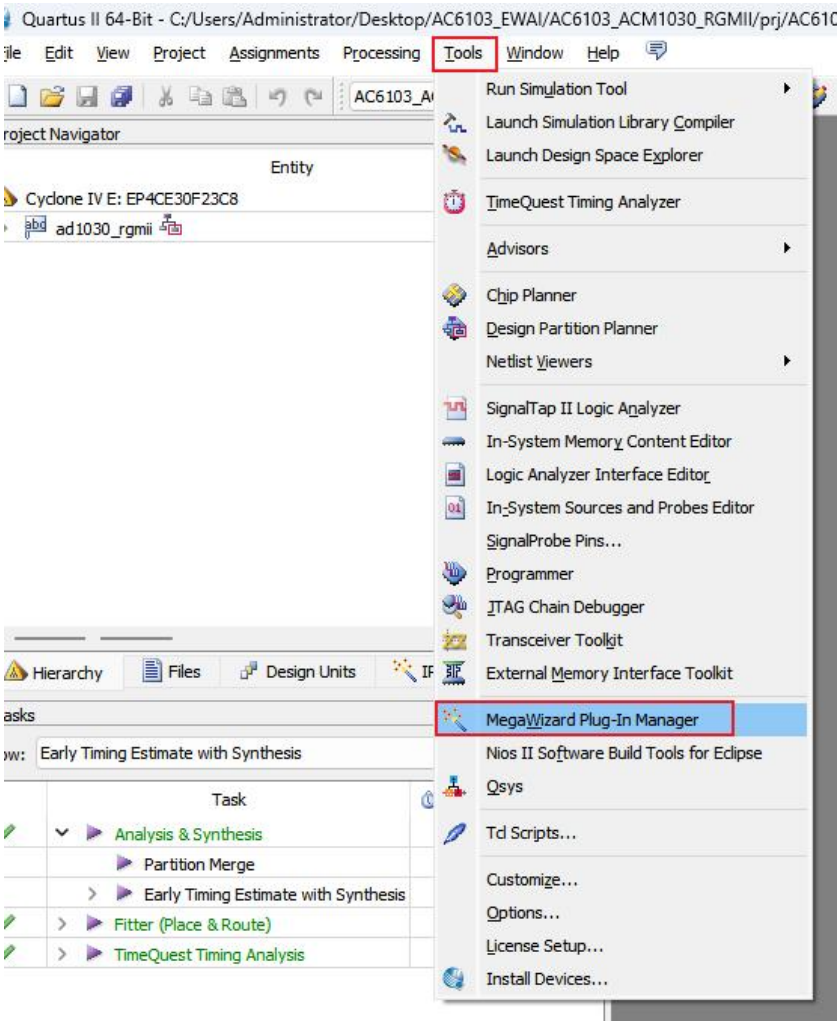

图 1-9 搜索 FIFO IP

双击 FIFO Generator 之后， 给 FIFO 命名为 fifo_tx，然后进入 FIFO 配置界面。

首先勾选使用不同输出位宽，由于 fifo_tx 的输入数据信号是由 ad1030_10bit_to_16bit 模块输出的 16 位的数据，而输出的数据需要交由以太网发送模块去输出，以太网发送模块输入的数据位宽是 8 位的，所以我们这里需要设置不同的位宽，深度我们设置为 16384，并勾选读写使用独立时钟，创建独立读写时钟是因为 ADC 驱动模块的工作时钟为 50M，而以太网发送模块工作时钟为 125M，写入和读出的时钟不一致，所以这里需要创建一个独立读写时钟 FIFO。配置界面如下图 1-10 所示。

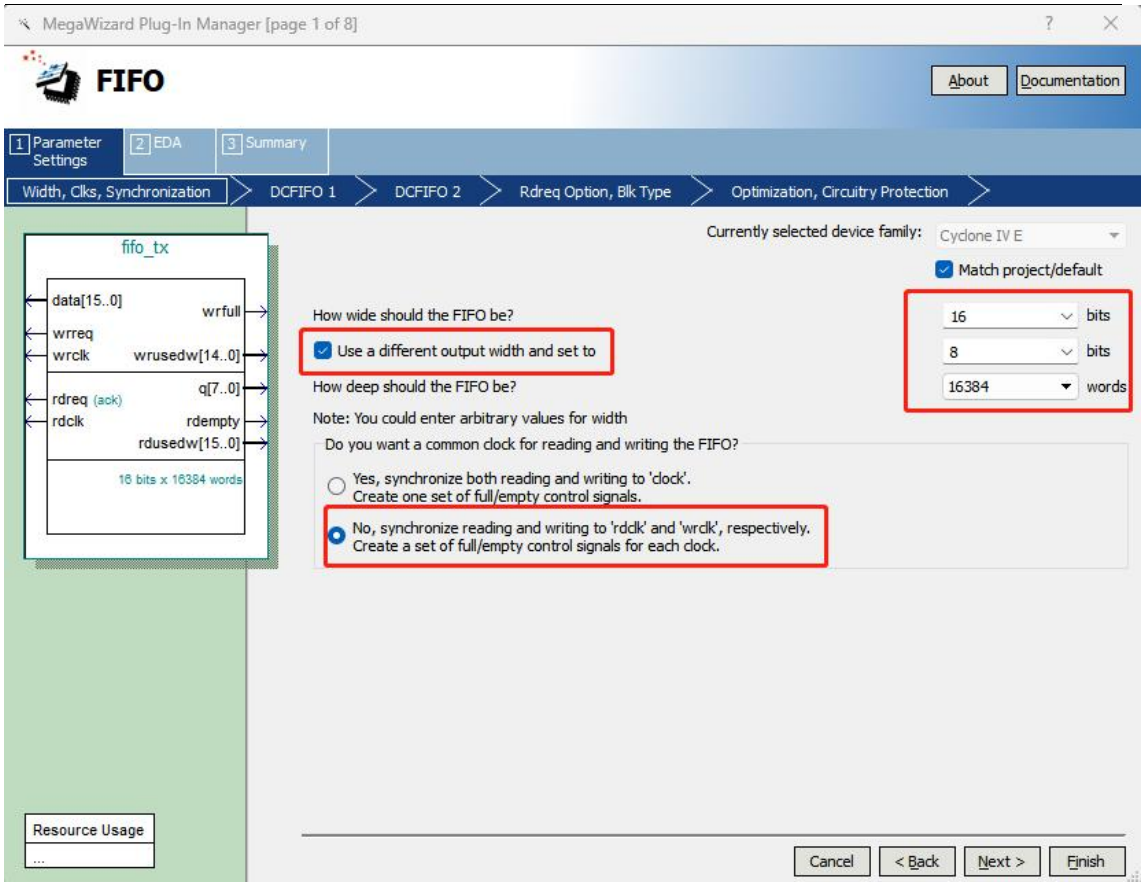

图 1-10 FIFO 配置界面 1

然后勾选读空，读计数，写满和写计数信号，方便其他模块使用这些信号，配置如下图 1-11 所示。

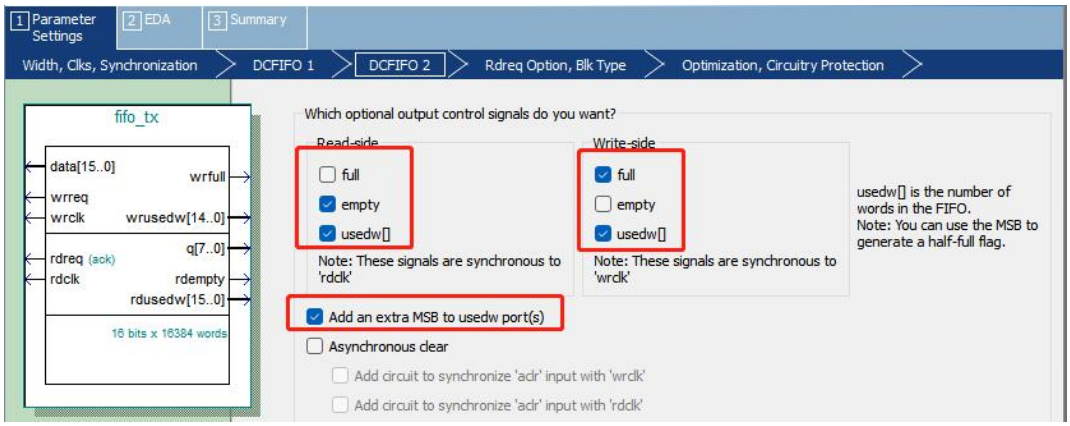

图 1-11 FIFO 配置界面 2

通过上述步骤，完成对 FIFO 的配置，这里只简单的对 FIFO 中的某些配置进行了说明，关于 FIFO 更详细的使用请读者查看前面“IP 核使用之 FIFO”一章的内容。

1.3.6 网口发送控制模块

网口发送控制模块（eth_send_ctrl）主要负责配置控制网口发送模块的使能控制信号pkt_tx_en，并通过pkt_length信号对以太网数据帧数据长度进行控制，该模块的结构框图如下图 1-12 所示。

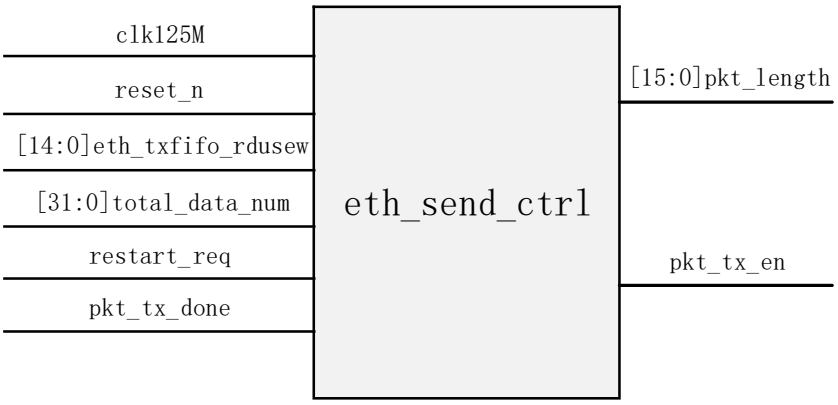

图 1-12 网口发送控制模块

对该模块的信号说明如下表 1-7 所示：

表 1-7 网口发送控制模块信号说明表

信号名称	I/O	信号意义
clk125M	I	模块时钟信号信号，以太网工作时钟 125M
reset_n	I	模块复位信号，低电平复位
eth_txfifo_rdusew [14:0]	I	FIFO 读数据计数
total_data_num[31:0]	I	需要采集的数据个数
restart_req	I	开始采样请求信号

pkt_tx_done	I	以太网一个包传输完成标志信号
pkt_length [15:0]	O	以太网需要传输的数据长度
pkt_tx_en	O	以太网传输使能信号

以太网帧最大长度 1518 字节（数据段 1500 字节），其中数据段 1500 字节还包括 20 字节 IP 报文头部和 8 字节 UDP 报文头部，所以数据帧发送的 ACM1030 采集的数据最大长度为 1472 字节。

该模块可以通过编写状态机代码的方式实现功能，下面将对每个状态的代码进行介绍。

状态 0，得到 pkt_length 信号的初始值。这里需要注意的是经过数据位扩展模块输出的数据为 16 位的，每个数据占据 2 个字节，所以发送 N 个采样数据，则以太网需要发送 2*N 个字节数据。当产生开始采样请求 restart_req 之后，系统开始采样，同时，将需要采集的数据个数 total_data_num 左移一位（相当于乘以 2），得到实际需要以太网传输的数据，当数据大于 16'd1472 时，设置 pkt_length 为最大传输长度 1472；当数据大于 0 时，pkt_length 等于为 total_data_num 乘以 2。给 pkt_length 赋初值之后，跳转到状态 1，代码如下所示：

```
if(restart_req)begin
  data_num <= total_data_num;
  if((total_data_num << 1) >= 16'd1472)begin
    pkt_length <= 16'd1472; //一个数据 2 个字节
    state <= 1;
  end
  else if((total_data_num << 1) > 0)begin
    pkt_length <= total_data_num << 1; //一个数据 2 个字节
    state <= 1;
    end
else begin
  state <= 0;
  end
end
```

状态 1，当 FIFO 计数信号 eth_txfifo_rdusew 的数值满足一帧数据帧发送采集数据长度，产生 pkt_tx_en 信号，以太网发送模块开始读取 FIFO 中的数据。代码如下所示：

```
if(eth_txfifo_rdusew >= (pkt_length -2))begin
  pkt_tx_en <= 1'b1;
  state <= 2;
end
else begin
  state <= 1;
  pkt_tx_en <= 1'b0;
end
```

状态 2，当以太网一个包传输完成之后，产生 pkt_tx_done 信号，此时剩下需要传输的数据 data_num 应该减去 pkt_length 的一半，这是因为 ADC 采集的数据是 16 位的，但是以太网每次只传送 8 位数据，所以以太网实际传输的数据应该 ADC 采集到的数据的一半。代码如下所示：

```
begin
  pkt_tx_en <= 1'b0;
  if(pkt_tx_done)begin
    data_num <= data_num  - pkt_length/2;
    state <= 3;
  end
end
```

状态3，设置以太网的帧间隙时间间隔。本次实验使用的千兆以太网，其相邻的两帧之间的最小间隔时间为 96ns，在设置的时候只要大于 96ns 就行，本次实验设置 128 个时钟周期，也就是 1024ns。当设置的时间比较小的时候，虽然以太网传输速率会加快，但是会增加电脑端解析数据包的压力，当时间过大，又会影响以太网传输速率，所以在设置的时候需要设定一个比较合理的值。

```
if(cnt_dly_time >= cnt_dly_min)begin
  state <= 4;
  cnt_dly_time <= 28'd0;
end
else begin
  cnt_dly_time <= cnt_dly_time + 1'b1;
  state <= 3;
end
```

状态 4，进行连续的包传输。当剩下的需要以太网传输的数据（data_num * 2）大于包传输最大数据 1472 时，使 pkt_length 为 1472，然后回到状态 1 继续传输；当剩下需要传输的数据不足包传输最大数据 1472 时， pkt_length 为剩下的需要传输的数据 data_num * 2，然后回到状态 1 传输剩下的数据。代码如下所示。

```
begin
  if(data_num * 2 >= 16'd1472)begin
    pkt_length <= 16'd1472;
    state <= 1;
  end
  else if(data_num * 2 > 0)begin
    pkt_length <= data_num * 2;
    state <= 1;
  end
  else begin
    state <= 0;
  end
end
```

模块设计完成之后，只需要在顶层文件中对各个模块之间的接口信号进行连接，完整的顶层文件代码请自行查看例程文件。

1.4 板级验证

经过以上工作，代码设计部分的任务已经全部完成，接下来就可以进行板级验证了。本次实验的板级验证环节，主要验证： 通过电脑上的网络调试助手，将命令帧进行发送， 然后通过 AC6103 开发板上的以太网芯片接收， 随后将接收到的数据转换命令， 最终实现对 ACM1030 模块的采样频率、数据采样个数以及采样通道的配置。配置完成之后，ACM1030 模块开始采集数据，将 ACM1030 模块采集的数据通过网口传输到电脑。电脑端将接收到的数据进行保存，然后通过 MATLAB 进行进一步的分析。针对本次实验，我们也提供有专门的上位机软件，用户只需要在软件界面进行参数配置， 便可以实时观察到实时的数据波形变化，使用起来非常方便。

1.4.1 硬件连接

将 ACM1030 模块、网线、下载器、电源线依次连接在开发板上，然后给 ACM1030 模块连接信号源，这里我们连接的是 ACM1030 的通道 1，信号源给的是 100Khz，vpp=5V 的正弦波，整体的硬件连接图如下图 1-13 所示。

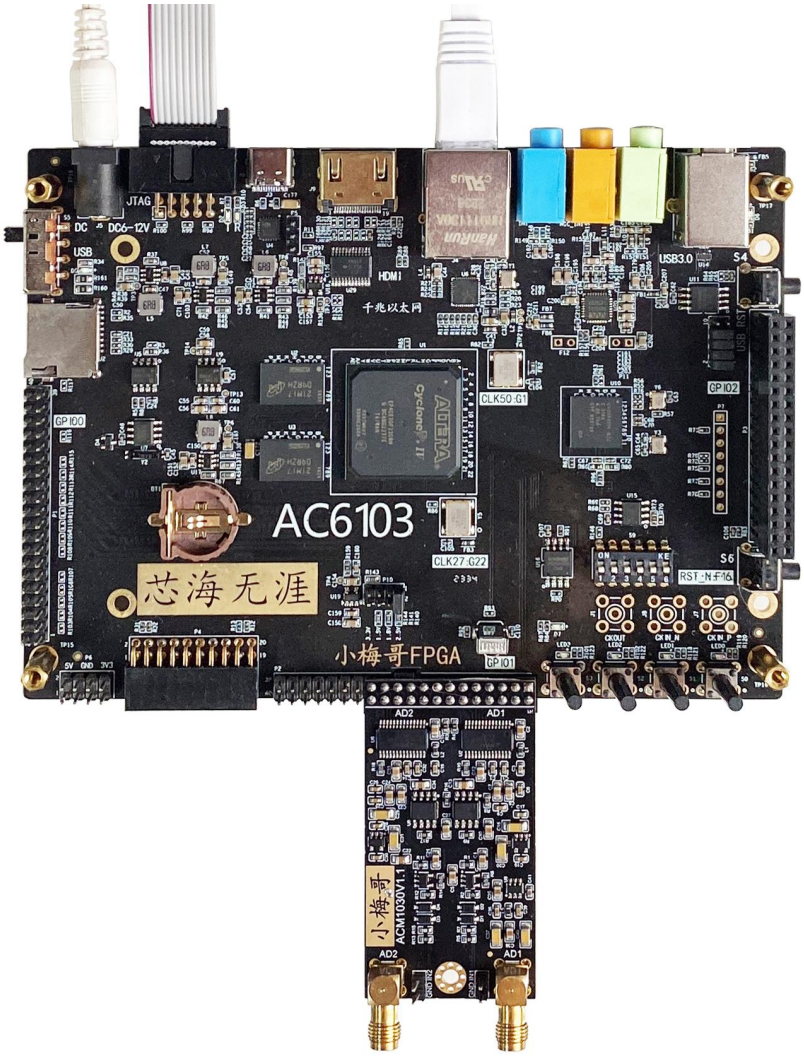

图 1-13 硬件连接图

连接完成之后， 将程序下载至开发板中， 然后就可以进行实验测试了。

1.4.2 修改电脑 IP 地址

本次实验设定了目标 IP 地址（PC 端）为 192.168.0.3，用户需要将自己电脑上的以太网 IP 地址修改为该地址，在本地连接状态中，点击属性，并在弹出的属性对话框中双击【Internet 协议版本 4 （TCP/Ipv4）】选项，然后在弹出的属性对话框中设置静态 IP 地址。 如下图 1-14 所示。

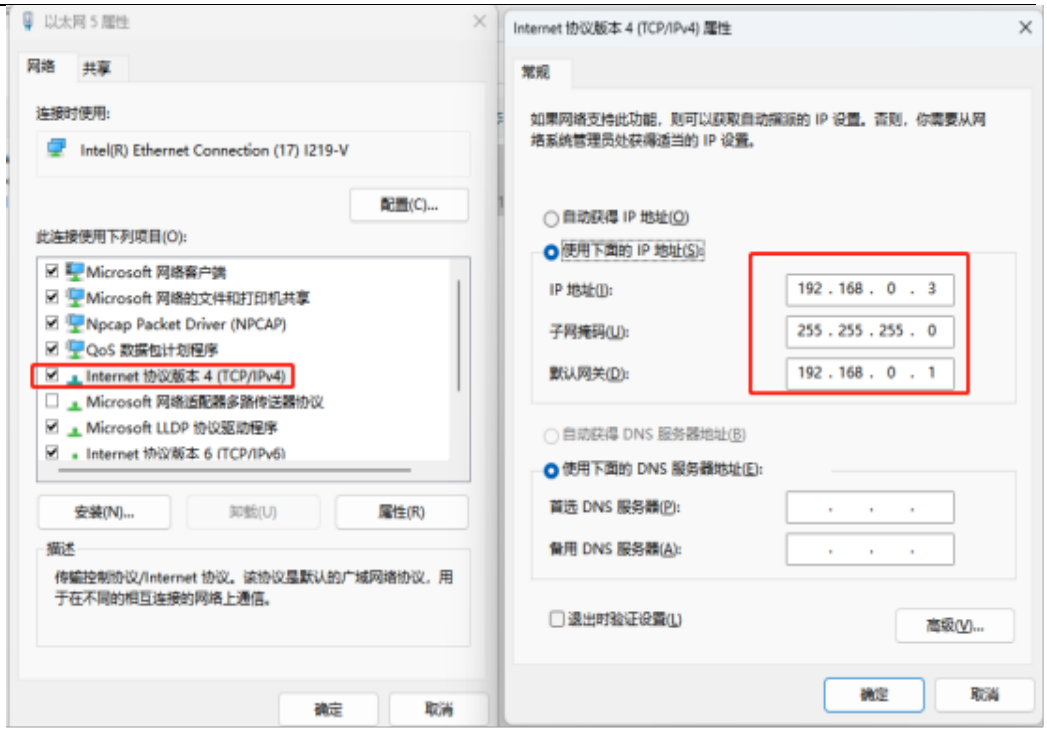

图 1-14 修改电脑 IP 地址

1.4.3 绑定 ARP

本工程不支持 ARP 协议，只能通过静态绑定的方式来强制将开发板的 IP 地址和 MAC 地址关联在一起。这样，当 PC 发送给 192.168.0.2 的数据包的时候，目标 MAC 地址自动为开发板的 MAC 地址。

关于 ARP的绑定请查看我们论坛上的帖子以太网通信静态 ARP 绑定方法与常见问题解决方案。

1.4.4 网络调试助手通信

完成上述操作之后，首先需要打开网络调试助手发送指令去配置，按照如下所述设置各项参数。网络调试助手软件读者可以在论坛中找到， 链接如下：

http://www.corecourse.cn/forum.php?mod=viewthread&tid=28374

1. 选择协议类型为 UDP。
2. 设置本地 IP 地址为 192.168.0.3。
3. 设置本地端口号为 6102。
4. 点击【连接】按钮以创建连接，连接上后该按钮为红色“断开”字样。
5. 连接上后， 设置目标主机为 192.168.0.2，目标端口为 5000。
6. 点击“接收保存到文件”这几个字，在弹出的界面中设置文件路径、

文件名称，如下图 1-15 所示。这样在数据接收完成之后会保存一个数 据文件。方便后面进行分析。

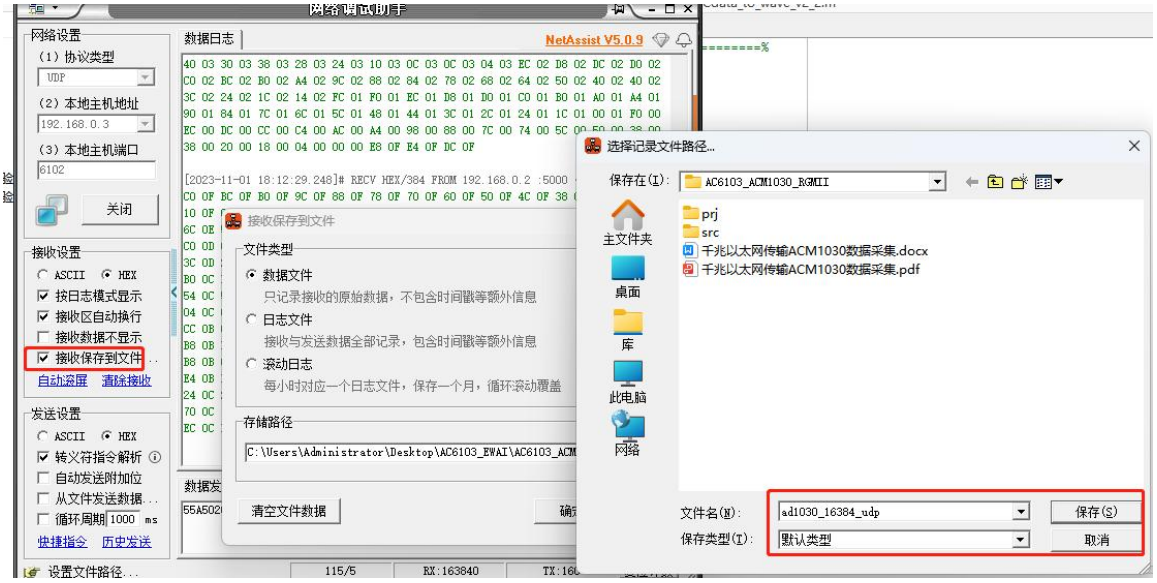

图 1-15 设置保存文件

在前面接收转命令模块中介绍到数据帧格式对 ACM1030 的四个寄存器的配置。

例如， PC 端要设置采样数据个数(DataNum 寄存器，地址为 2)为 16384(0x4000)个，发送数据帧内容： 0x55 0xA5 0x02 0x00 0x00 0x40 0x00 0xF0。PC 端要设置采样速率(ADC_Speed_Set 寄存器，地址为 3)为 50M，则需要发送的数据帧内容为： 0x55 0xA5 0x03 0x00 0x00 0x00 0x00 0xF0。

当上述设置都设置完成后，就可以向 0 号寄存器写入任意值， 来开始一次采样传输了。数据帧内容可以为 0x55 0xA5 0x00 0x00 0x00 0x00 0x00 0xF0，这里需要注意的是 0 号寄存器必须放在最后，因为 0 号寄存器负责启动 ADC，ADC 在未配置完全的情况下开始启动，数据很容易输出错误值。

开始传输数据帧命令发送完成之后，ACM1030 就能实现以 50M 的采样速率，对 1 个通道进行采样（本次实验以通道 1 为例），共采集 16384 个数据。四个寄存器对应的配置如下表 1-8 所示：

表 1-8 ACM1030 数据帧格式配置表

寄存器名称	数据帧数据
DataNum	55 A5 02 00 00 40 00 F0
ChannelSel	55 A5 01 00 00 00 01 F0

ADC_Speed_Set	55 A5 03 00 00 00 00 F0
RestartReq	55 A5 00 00 00 00 00 F0

配置成网络调试助手发送的数据格式如下：

55A50200004000F055A50100000001F055A50300000000F055A50000000000F0

最终的网络助手配置界面如下图 1-16 所示：

图 1-16 网络调试助手配置界面

点击发送之后可以看到网络调试助手在不断的接收数据，如下图 1-17 所示。从图中可以看出一共接收到 32768 个数据，这是因为设置的 ADC 采样数量为 16384，ADC 采样数据是 16 位的，以太网是以字节（8 位）为单位进行发送的，所以通过以太网接收到字节数应该是 16384*2 个数据，这也就是说明接收到的数据的个数没错。

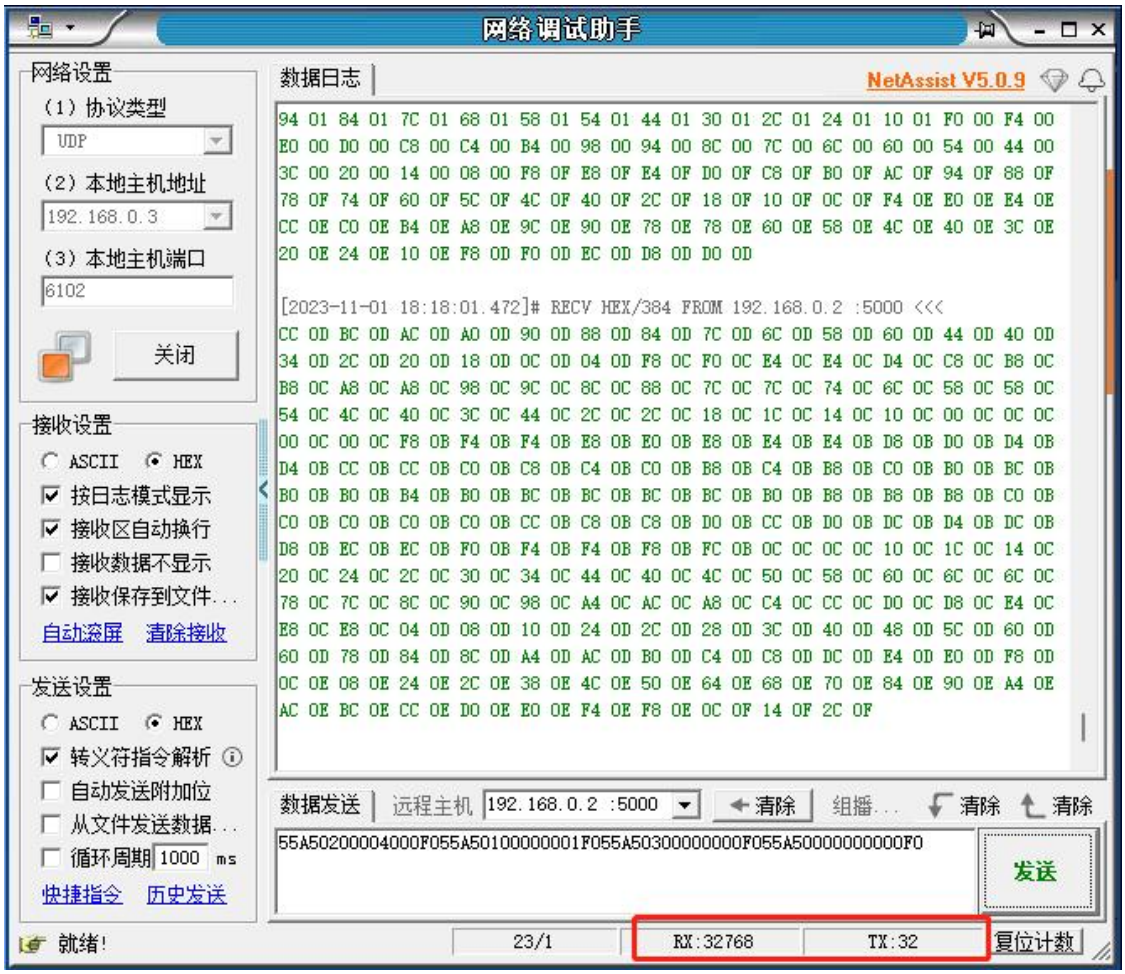

图 1-17 网络调试助手接收数据

1.4.5 MATLAB 图像绘制

前面通过网络调试助手得到了 ADC 采集到的数据文件，然后我们就需要对采集到的数据进行分析，本次实验使用MATLAB 软件进行分析。使用MATLAB 软件需要读者电脑安装了 MTALAB，如果已经安装好了 MTALAB 软件，则可以双击我们提供的 ADCdata_to_wave_v2_2.m 文件，在打开方式里选择以 MATLAB 打开， 将工程压缩包解压便可以看到该文件。文件打开之后，读者需要将代码中文件路径修改为你保存的数据文件路径，随后点击运行便可以直观的看到数据是否正确， MATLAB 操作如下图 1-18 所示，得到的波形图如下图 1-19 所示。

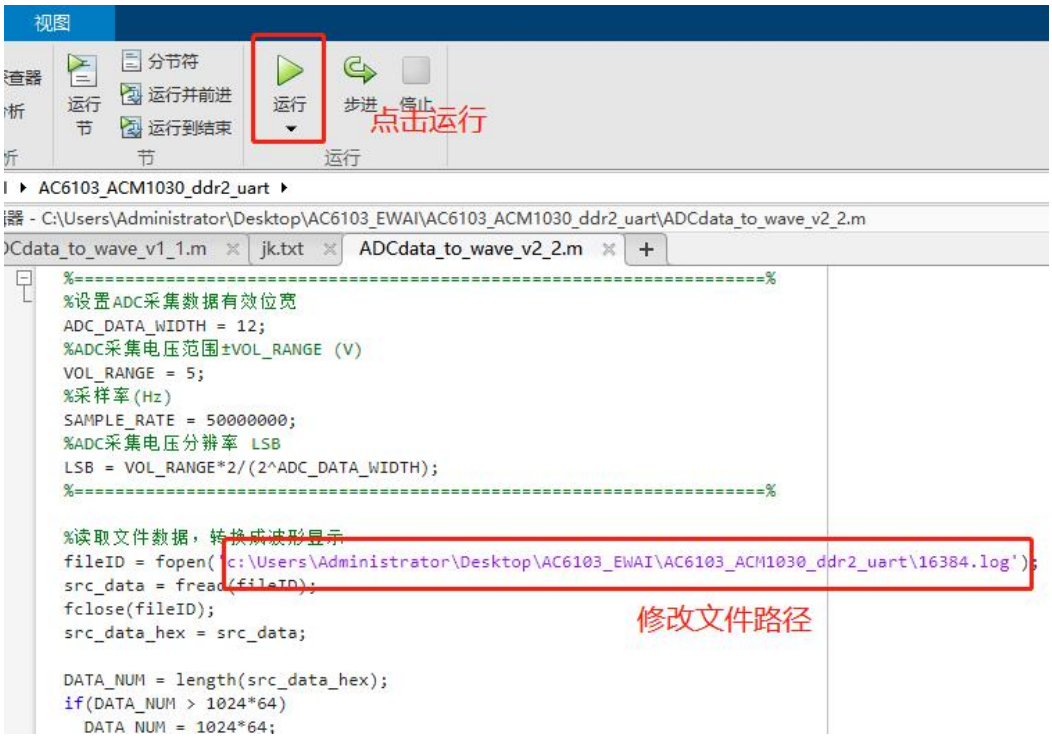

图 1-18 修改文件路径并运行

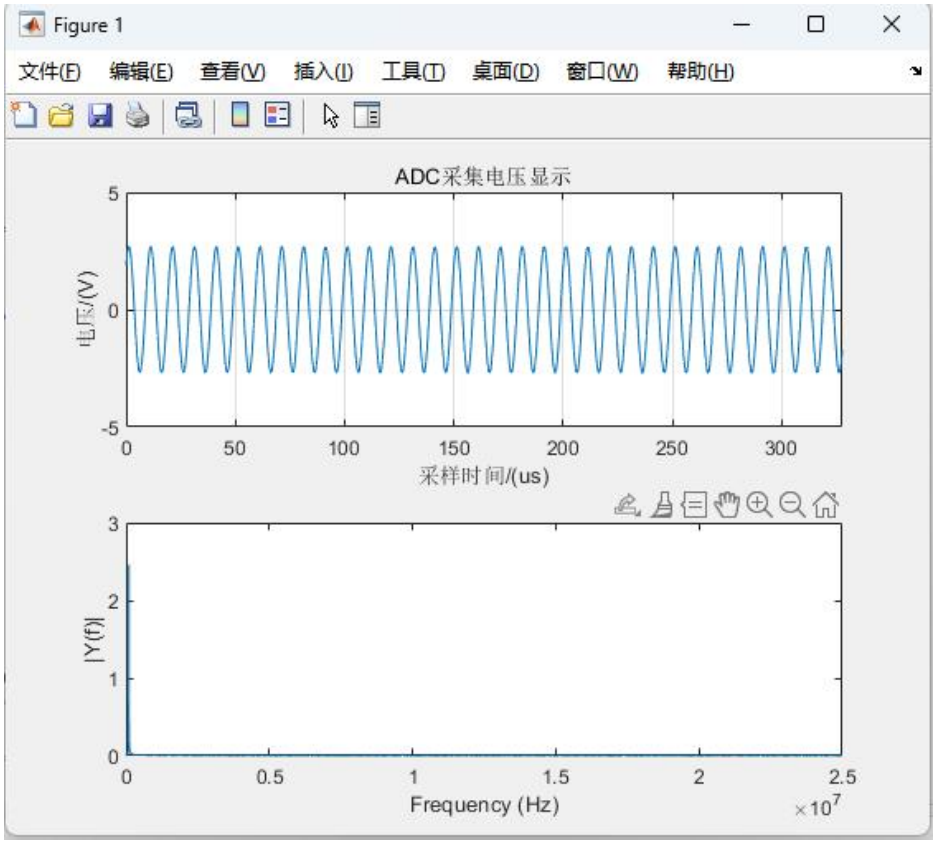

图 1-19 MATLAB 分析波形图

前面我们提到过本次实验提供的信号源为 100Khz，Vpp 为 5V 的正弦波（正负 2.5V），与 MATLAB 分析出来的波形一致，说明我们本次实验成功。

1.4.6 数据采集上位机通信

前面通过网络调试助手采集数据时，每次保存数据都需要重新点击“接收”保存文件”一栏，修改寄存器参数的时候，都需要重新计算，然后发送命令，修改之后也不能直接实时观察到数据波形，使用起来不是很方便。基于上述问题，我们设计了上位机软件“小梅哥控制台 For ADC 采集”进行数据采集，上位机内部直接对命令进行了构建，用户只需要在界面上对采样参数进行设置，便可以实时观测到数据变化，读者在论坛上搜索数据采集上位机便可以看到双击上位机软件，初始界面如下图 1-20 所示。

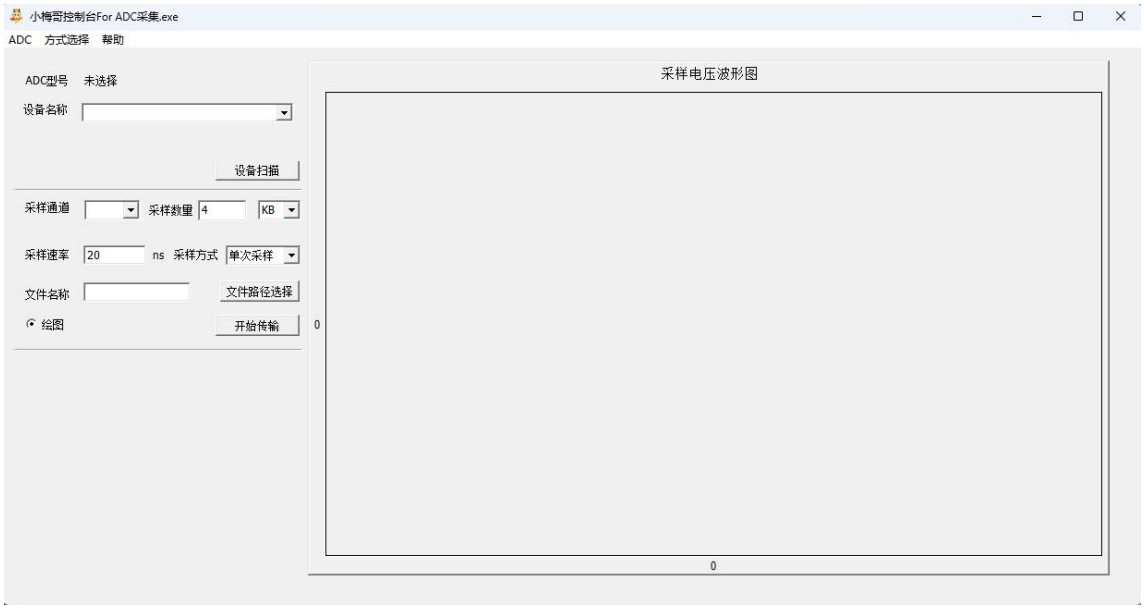

图 1-20 初始化界面

本次实验使用该软件的方式如下所示：

1. 点击 ADC，选择 ACM1030。
2. 点击方式，选择网口，可以看到主机 IP（PC 端）和目的 IP（FPGA） 以及对应的端口号。主机 IP：192.168.0.2，主机端口号： 6102；目的 IP：192.168.0.3，目的端口号： 5000。
3. 选择完成之后， 我们可以看到采样通道、采样数量等都已经设置了初 始值（默认设置的采样率为ADC 模块的最大采样率），用户可以根据自己的需求进行修改。
4. 点击网络连接。
5. 点击开始传输之后，可以看到在右边采样电压波形图界面可以直观看 到波形图，如下图 1-21 所示。需要注意的是波形图的横坐标对应的不 是频率，而是采样数量。

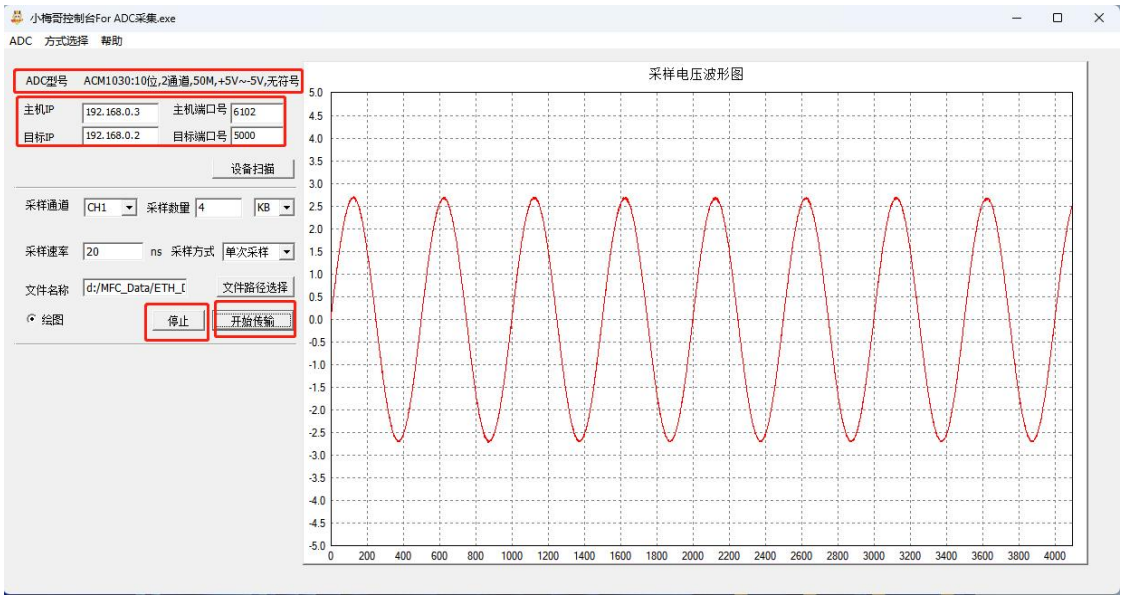

图 1-21 数据采集上位机显示图

通过上位机采集到的数据文件保存在 d:/ MFC_Data 文件夹下，后续可以通过 MATLAB 软件进行进一步的分析，通过 MATLAB 分析的波形图如下图 1-22 所示。从图中可以看出，采集到的数据是频率为 100Khz，电压在正负 2.5V 左右的正弦波，与我们输入的信号一致。

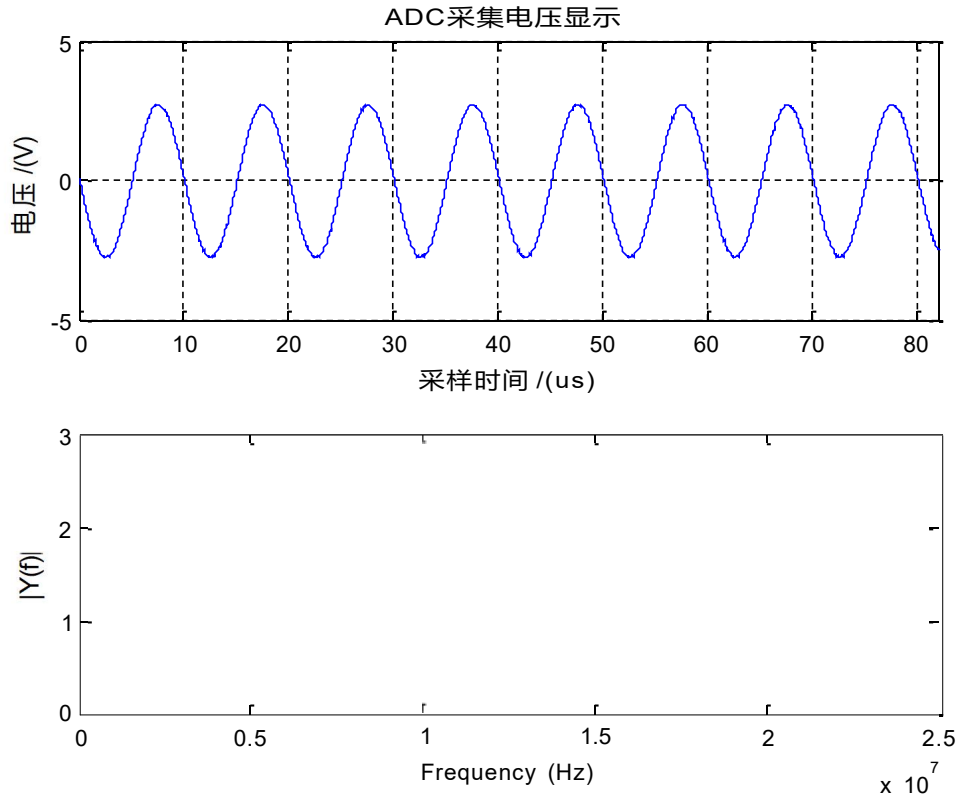

图 1-22 MATLAB 进一步波形分析图

1.5 思考与总结

本次实验介绍了基于 ACM1030 的千兆以太网收发，用户通过网口调试助手以太网帧向开发板发送指令数据配置 ACM1030 的四个寄存器，以此控制 ADC 进行采样，并将数据缓存后再组成以太网帧， 经由网口发送至电脑， 借由网口调试工具对数据进行查看。 如果使用我们提供的上位机软件， 则不需要自己设置命令，只需要在界面上修改相关参数，便可以在右边的波形显示界面实时观察到波形变化。

实验中需要注意的是在配置寄存器的过程中要考虑到 FIFO 的写深度，一次采样所能采集的数据应该小于或等于 FIFO 的写深度，同时负责启动 ADC 的 0 号寄存器应该放在代码指令的最后进行配置。