1 ACM1030 数据采集 DDR3 缓存网口发送

工程源码	|-- ACG525 开发板 |-- acg525_ad1030_ddr3_rgmii
相关视频课程	

章节导读

本实验将介绍基于 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 DDR3 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 ACG525 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 DDR 中。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。

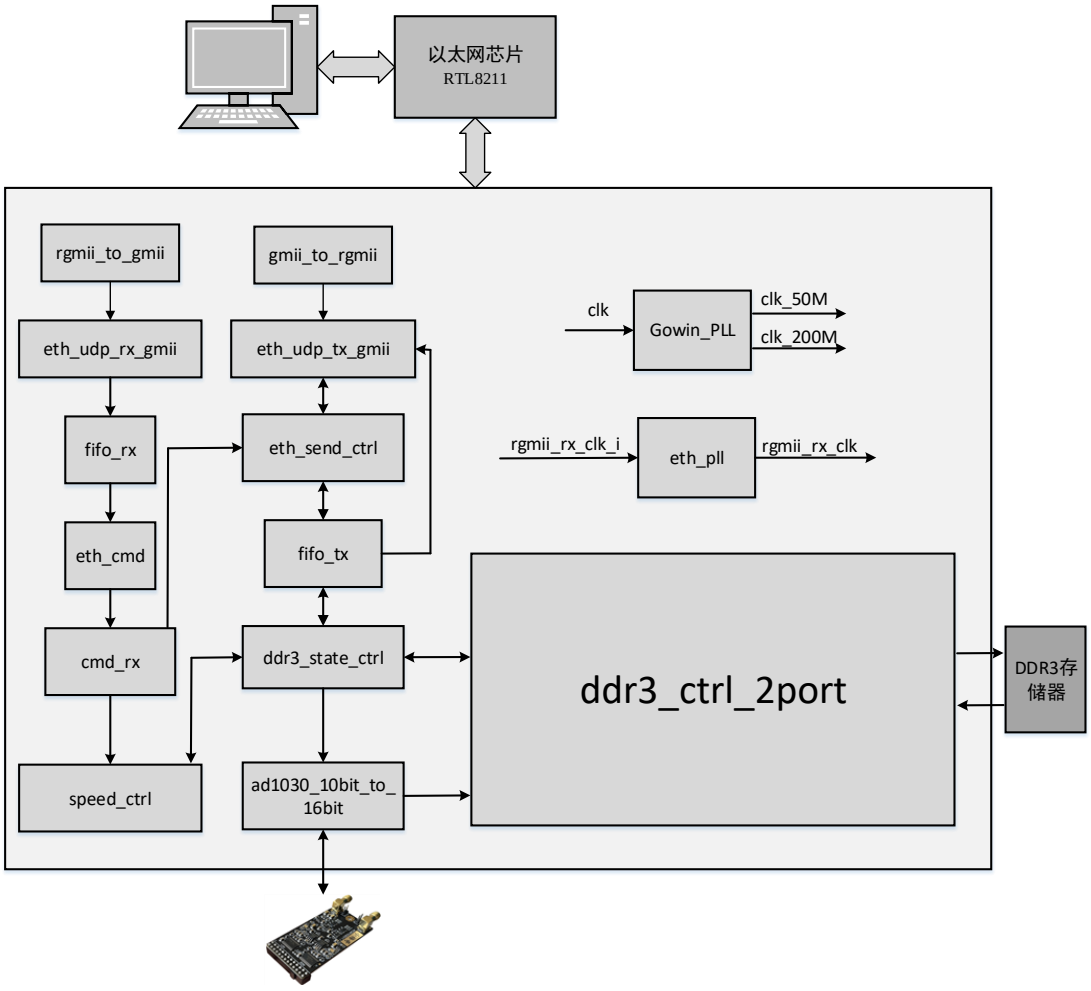

图 1-1 ACM1030 数据采集 DDR3 缓存网口整体设计框图

对于上图中各个模块的功能介绍如下：

1. Gowin_PLL 模块：锁相环模块，输入时钟 50M，由开发板上的晶振提供；输出 200M 的时钟给到 DDR3 控制器使用；输出 50M 的时钟给其他模块使用。

2. eth_pll 模块：锁相环模块，将 rgmii 接口时钟信号 rgmii_rx_clk_i 偏移 270°得到 rgmii_rx_clk 时钟信号，这样做是为了在时钟上升沿/下降沿取数据时，取得的数据是最准确和稳定的，具体的偏移角度是实际使用时，不断调试得到的，在我们的高云的开发板上，偏移 270°是最合适的。

3. rgmii_to_gmii 模块：以太网接收 rgmii 转 gmii 模块。

4. 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。

5. 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。
6. 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。
7. 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。
8. speed_ctrl 模块：采样速率控制模块，控制 ACM1030 的采样速率。
9. ad_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。
10. state_ctrl 模块： ADC 采集数据 DDR3 缓存以太网发送状态控制模块，协调各个模块的信号控制，程序状态的总控制模块。
11. fifo_tx 模块：FIFO IP 核，DDR3 的双端口模块中的 FIFO IP 的工作时钟与 ACM1030 模块的工作时钟一致为 50M，而以太网发送模块的工作时钟为 125M，两者数据速率不匹配，使用该 FIFO IP 核解决采集过程中会出现的跨时钟域数据交互问题。FIFO 还可以将从 DDR3 双端口模块中读取出来的 16 位数据转换成 8 位数据，供以太网发送模块进行发送。
12. eth_send_ctrl 模块：以太网发送控制模块，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。
13. 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。
14. 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。

本次实验需要设计的模块包括eth_cmd 模块、cmd_rx 模块、speed_ctrl 模块、ad1030_10bit_to_16bit 模块、state_ctrl 模块和 eth_send_ctrl 模块，关于以太网发送\接收模块和 DDR3 双端口模块的设计本次实验将不做介绍，相关内容可以查看开发板文档中的相关章节。

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。

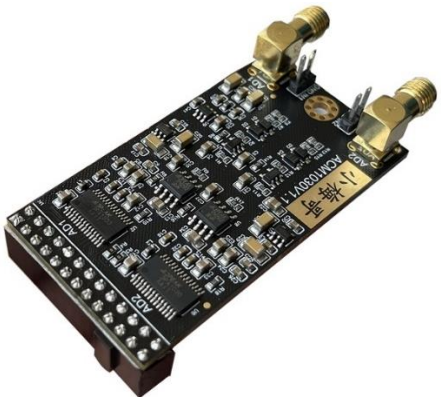

图 1-2 ACM1030 模块图

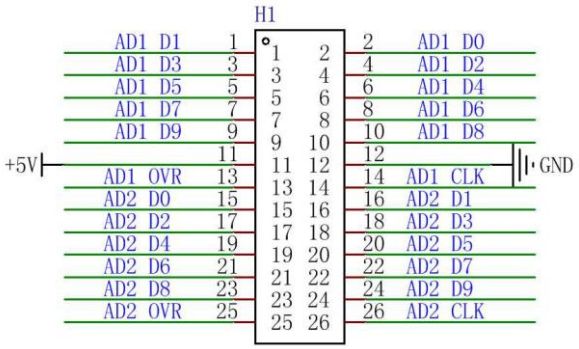

图 1-3 ACM1030 模块接口图

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。

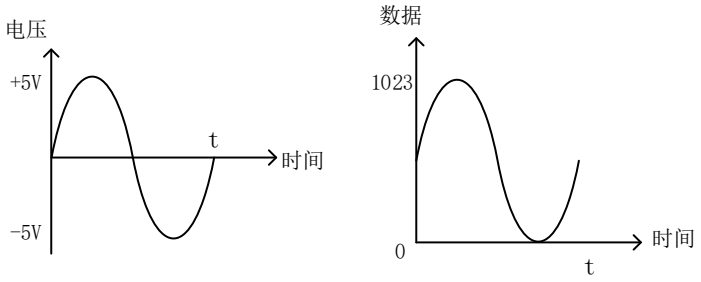

图 1-4 3PA1030 正弦波模拟电压值（左）、数据（右）

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、AC6102、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。

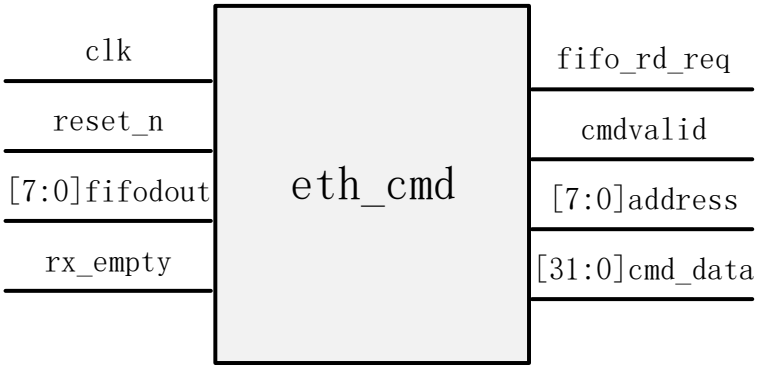

图 1-5 接收转命令模块框图

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。

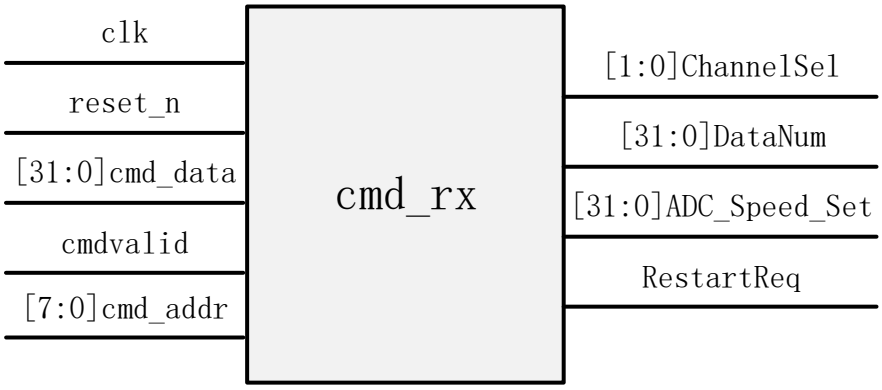

图 1-6 指令转控制模块结构框图

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。

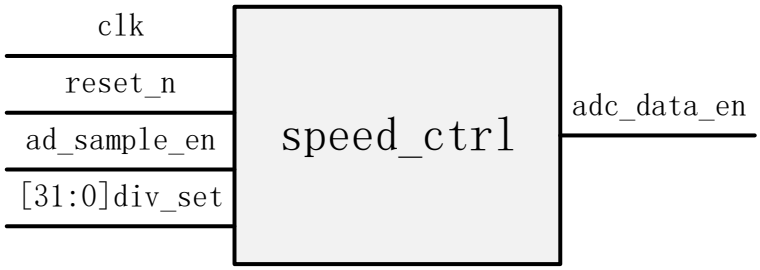

图 1-7 采样速率控制模块

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_en	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
adc_data_en	O	ADC 采样结果存储使能信号

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。下面我们将编写相应代码实现上述功能。

设置一个计数器 div_cnt，当产生采样使能信号 ad_sample_en 之后，计数器加 1，当计数值等于设置的 div_set 的时候，将计数器清零。代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
    div_cnt <= 0;
else if(ad_sample_en)begin
    if(div_cnt >= div_set)
        div_cnt <= 0;
    else
        div_cnt <= div_cnt + 1'd1;
end
else
div_cnt <= 0;
```

计数器的计数值达到 div_set 的时候，使能 ADC 采样结果存储使能信号 adc_data_en，我们将该信号输出，最终实现每隔 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：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
    adc_data_en <= 0;
else if(div_cnt == div_set)
    adc_data_en <= 1;
else
adc_data_en <= 0;
```

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（ad_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。

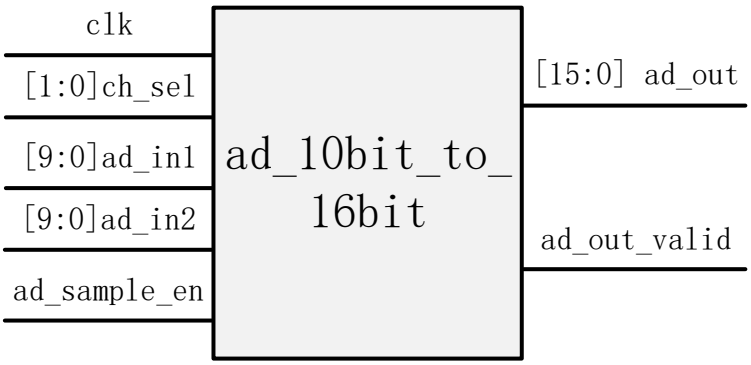

图 1-8 数据位扩展模块结构框图

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
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_sample_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_sample_en 信号给到 ad_out_valid，代码如下所示：

```
always @(posedge clk)
    ad_out_valid <= ad_sample_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)
if(ad_sample_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_sample_en && ch_sel == 2'b10)
    ad_out<={4'd0,s_ad_in2,2'd0};//
else if(ad_sample_en && ch_sel == 2'b00)
    ad_out<={4'd0,adc_test_data,2'd0};
else
    ad_out <= 16'd0;
```

1.3.5 DDR3 缓存状态控制模块

DDR3 双端口控制器的控制信号的产生以及 ADC 何时启动数据传输都是通过 state_ctrl 控制的，该模块的结构框图如下图 1-9 所示。

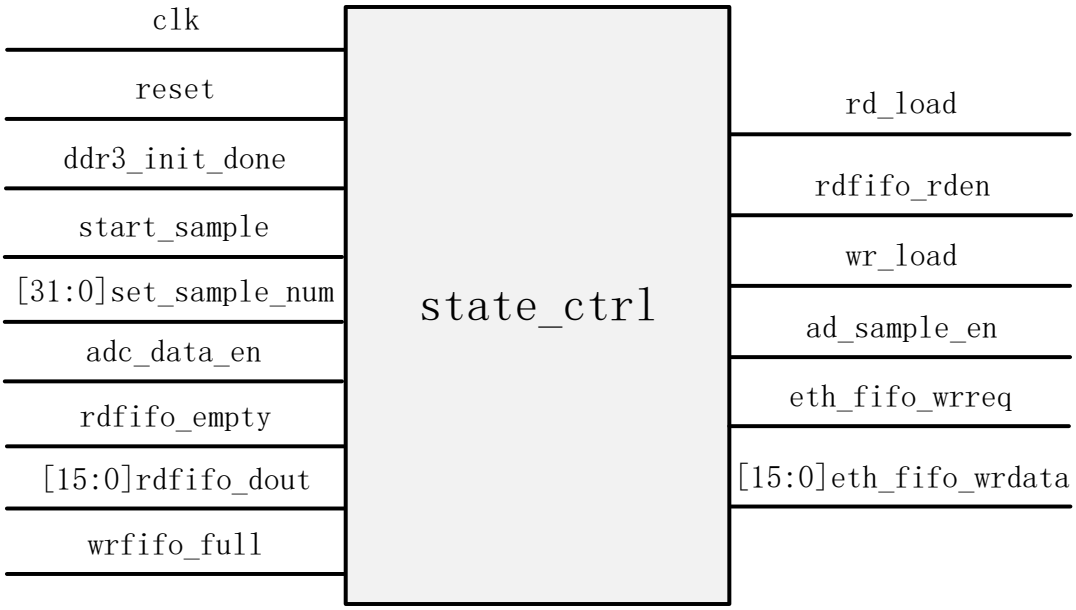

图 1-9 state_ctrl 模块结构框图

对于该模块的信号说明如下表 1-7 所示。

表 1-7 state_ctrl 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，与 ADC 采集模块的时钟信号保持一致（50M）
reset	I	模块复位信号，高电平复位
ddr3_init_done	I	DDR3 初始化完成标志信号
start_sample	I	ADC 模块开始采样标志信号
set_sample_num [31:0]	I	设置的采样个数
adc_data_en	I	ADC 采样结果存储使能信号
rdfifo_empty	I	读 FIFO 的读空标识信号，用于标识当前 FIFO 是否为空（即 FIFO 内有

		无数据）
rdfifo_dout[15:0]	I	读 FIFO 的数据输出，数据位宽为 16 位
wrfifo_full	I	写 FIFO 的写满标识信号，用于标识当前 FIFO 是否有被写满
rd_load	O	DDR3 双端口模块读出源更新信号
rdfifo_rden	O	读 FIFO 的读数据使能控制信号，给高电平表示往 FIFO 读数据，为避免读数据的丢失，确保在 FIFO 非空（rdfifo_empty =0）情况下读数据
wr_load	O	DDR3 双端口模块写入源更新信号
ad_sample_en	O	ADC 采样使能标志信号
eth_fifo_wrreq	O	以太网发送缓存 FIFO 的写请求信号
eth_fifo_wrdata [15:0]	O	写入以太网发送缓存 FIFO 的 16 位数据

DDR3 双端口模块（ddr3_ctrl_2port）需要的控制信号有：rd_load、wr_load、wrfifo_clk、wrfifo_wren、wrfifo_din、rdfifo_clk、rdfifo_rden。

上述信号中：wrfifo_clk、rdfifo_clk 应该与该模块的工作时钟保持一致，也就是和 ADC 数据采集模块的工作时钟保持一致为 50M；wrfifo_wren 信号由 ADC 数据位扩展模块输出数据有效信号 ad_out_valid 与输出数据使能信号相与得到；wrfifo_wren 信号由 ADC 模块输出数据有效信号 ad_out_valid。除去上述已经得到的控制信号外， state_ctrl 模块还需要产生的控制信号包括：rd_load、wr_load、rdfifo_rden。

根据上述描述，我们可以通过状态机实现该模块的功能。定义状态如下所示。

表 1-8 state_ctrl 状态控制模块状态描述表

状态值	状态名称	状态意义
0	IDLE	空闲状态
1	DDR_WR_LOAD	DDR 写入数据更新状态
2	ADC_SAMPLE	ADC 采样数据状态
3	DDR_RD_LOAD	DDR 读出数据更新状态
4	DATA_SEND_START	数据发送状态
5	DATA_SEND_WORKING	数据发送完成状态

下面编写每个状态对应的代码。

1. IDLE 状态

当处于空闲状态并且 DDR3 初始化完成之后，产生启动传输开始信号 start_sample_rm。代码如下所示：

```
always@(posedge clk or posedge reset)begin
if(reset)
    start_sample_rm <= 1'b0;
else if(state==IDLE && ddr3_init_done==1'b1)
    start_sample_rm <= start_sample;
else
```

```
    start_sample_rm <= 1'b0;
end
```

当产生 start_sample_rm 信号之后跳转到 DDR_WR_LOAD 状态。空闲状态代码代码如下所示：

```
begin
    if(start_sample_rm) begin    //DDR 初始化完成并且产生启动采样信号
        state <= DDR_WR_LOAD;
    end
    else begin
        state <= state;
    end
end
```

2. DDR_WR_LOAD 状态

进入 DDR 写入数据更新状态之后，开始清除写入 DDR 内的原始数据。设置清除 DDR 内数据的计数器，经过验证，进行 10 拍的延时之后，能够完全清除之前存储在 DDR 内的数据，不影响下一次的传输，代码如下所示：

```
always@(posedge clk or posedge reset)begin
if (reset)
    wr_load<=0;
else if(ddr3_init_done==1'b0)
    wr_load<=1'b1;
else if(state==DDR_WR_LOAD)
    begin
        if(wr_load_cnt==0||wr_load_cnt==1||wr_load_cnt==2)
            wr_load<=1'b1;
        else
            wr_load<=1'b0;
    end
else
    wr_load<=1'b0;
end
```

然后等待 wrfifo_full（写端 fifo 满信号）的信号拉低，拉低后，表示可以往 FIFO 里写入数据，此时进入下一个状态。在清空（复位）FIFO 的时候，FIFO 的 full 信号会变高，可以认为在复位 FIFO 时是不允许对 FIFO 进行写操作的，即使写也是不可靠的，等 FIFO 的复位结束后，full 信号会变低，就允许对 FIFO 进行写操作。DDR 写入数据更新状态代码如下所示：

```
begin
    if(!wrfifo_full && (wr_load_cnt==9))
        state<=ADC_SAMPLE;
    else
        state<=state;
```

```
end
```

当处于 DDR_WR_LOAD 状态时，我们需要产生 wr_load 信号，由三拍延时信号拉高提供，之所以提供延迟信号时间为 3 拍，是为了能够准确清除已经写入的数据，代码如下所示。

```
always@(posedge clk or posedge reset)begin
if (reset)
    wr_load<=0;
else if(ddr3_init_done==1'b0)
    wr_load<=1'b1;
else if(state==DDR_WR_LOAD)
    begin
        if(wr_load_cnt==0||wr_load_cnt==1||wr_load_cnt==2)
            wr_load<=1'b1;
        else
            wr_load<=1'b0;
    end
else
    wr_load<=1'b0;
end
```

3. ADC_SAMPLE 状态

进入 ADC 采样数据状态之后，首先设置 ADC 采样个数计数器 adc_sample_cnt，当产生 adc_data_en 信号之后，我们就将 adc_sample_cnt 计数值加 1，对 ADC 采集的数据进行计数。代码如下所示：

```
always@(posedge clk or posedge reset)
if(reset)
    adc_sample_cnt<=32'd0;
else if(state==ADC_SAMPLE)begin
    if(adc_data_en)
    adc_sample_cnt<=adc_sample_cnt+1'b1;
    else
    adc_sample_cnt<=adc_sample_cnt;
end
else
  adc_sample_cnt<=1'b0;
```

当 adc_sample_cnt 达到设定的采样数据个数，并且 adc_data_en 有效的时候时，ADC 模块数据采集完成，跳转到读出数据更新状态，代码如下所示：

```
begin
    if((adc_sample_cnt>=set_sample_num-1'b1)&& adc_data_en)
        state<=DDR_RD_LOAD;
    else
        state<=state;
end
```

当处于 ADC_SAMPLE 状态时，我们还需要产生采样使能信号 ad_sample_en 给到其他模块使用，代码如下所示：

```
always@(posedge clk or posedge reset)begin
if(reset)
    ad_sample_en<=0;
else if(state==ADC_SAMPLE)
    ad_sample_en<=1;
else
    ad_sample_en<=0;
end
```

4. DDR_RD_LOAD 状态

进入 DDR 读出数据更新状态之后，首先设置读取数据更新状态计数器 rd_load_cnt，设置 10 拍的延时，代码如下所示：

```
always@(posedge clk or posedge reset)begin
if(reset)
    rd_load_cnt<=0;
else if(state==DDR_RD_LOAD)
begin
    if(rd_load_cnt==9)
        rd_load_cnt<=4'd9;
    else
        rd_load_cnt<=rd_load_cnt+1'b1;
end
else
    rd_load_cnt<=1'b0;
end
```

然后等待 rdfifo_empty（读端 FIFO 的空信号）信号拉低，拉低后，表示 FIFO 里已经有被写入数据，此时进入下一状态。在清空（复位）FIFO 的时候，FIFO 的 empty 信号会变高，可以认为在复位 FIFO 时是不允许对 FIFO 进行读操作的，即使读也是不可靠的，等 FIFO 的复位结束后，FIFO 被写入数据后，empty 信号会变低，就允许对 FIFO 进行读操作，然后跳转到 DATA_SEND_START 状态，DDR_RD_LOAD 状态对应的代码如下所示：

```
begin
    if(!rdfifo_empty && (rd_load_cnt==9))begin
        state<=DATA_SEND_START;
    end
    else
        state<=state;
end
```

当处于 DDR_RD_LOAD 状态时，我们需要产生 rd_load 信号，有三拍的延

时信号拉高提供，之所以提供的延迟信号时间为 3 拍，是为了保证 rd_load 指令的可靠，代码如下所示：

```
always@(posedge clk or posedge reset)begin
if (reset)
    rd_load<=0;
else if(ddr3_init_done==1'b0)
    rd_load<=1'b1;
else if(state==DDR_RD_LOAD)
begin
    if(rd_load_cnt==0||rd_load_cnt==1||rd_load_cnt==2)
        rd_load<=1'b1;
    else
        rd_load<=1'b0;
end
else
    rd_load<=1'b0;
end
```

5. DATA_SEND_START

进入DATA_SEND_START 状态之后，state 直接跳转到数据发送状态，USB 启动传输，代码如下所示：

```
begin
    state <= DATA_SEND_WORKING;
end
```

6. DATA_SEND_WORKING 状态

进入数据发送状态之后，产生 rdfifo_rden 信号，将 DDR 中的数据读取出来，当读出的数据达到需要采集的数据信号时，跳转到 IDLE 状态，完成一次数据传输，代码如下所示：

```
begin
    if(send_data_cnt >= set_sample_num-1'b1) begin
        rdfifo_rden <= 1'b0;
        state <= IDLE;
    end
    else begin
        rdfifo_rden <= 1'b1;
        state <= DATA_SEND_WORKING;
    end
    end
```

send_data_cnt 在 rdfifo_rden 有效的情况下进行计数，从而统计从 DDR 中读取的数据个数，代码如下所示：

```
always@(posedge clk or posedge reset)begin
```

```
if(reset)
    send_data_cnt<=32'd0;
else if(state==IDLE)
    send_data_cnt<=32'd0;
else if(rdfifo_rden)
    send_data_cnt<=send_data_cnt+1;
else
    send_data_cnt<=send_data_cnt;
end
```

1.3.6 FIFO 数据存储模块

FIFO 双端口 IP 核（fifo_tx），该模块需要接收从 DDR 读出的 16 位数据，数据经由 fifo_tx 缓存后转换成 8 位数据由以太网发送模块发送出去。ddr3_ctrl_2port 模块中 FIFO IP 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过程中会出现的跨时钟域数据交互问题。

在 GOWIN 软件中点击  ，然后在搜索栏中输入 FIFO，在下面搜索结果中找到 FIFO 并双击，操作如下图 1-10 所示。

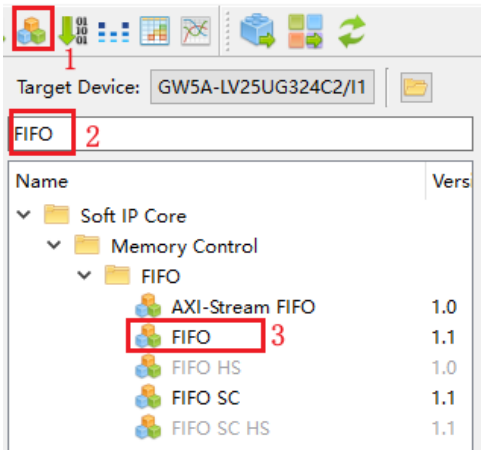

图 1-10 搜索 FIFO IP

双击 FIFO Generator 之后，进入 FIFO 配置界面。

首先将 File Name 和 Module Name 的名称修改为 fifo_tx，然后按照如下步骤进行配置：

1. 写数据位宽设置为 16，写入深度设置为 1024。因为 ADC 模块输出的数据是 16 位的，所以位宽设置为 16。写入深度用户可以修改，理论上只要大于以太网最大帧长度 1472（写入深度大于 1472/2）就可以。

2. 读出位宽设置为 8。设置为 8 是因为以太网发送模块的数据位宽是 8 位的。

3. 读模式选择为“First-Word Fall-Through（FWFT）”，FWFT 模式可以不需要读命令，自动将最新的数据放在 Q 上。

4. 勾选“Read Data Num”和“Write Data Num”。FIFO 数据量计数信号输出，Read Data Num 和 Write Data Num 分别同步于写时钟和读时钟。

5. 勾选“En_Reset”和“Reset_Synchronization”，使能同步复位。

最终配置界面如下图 1-11 所示。

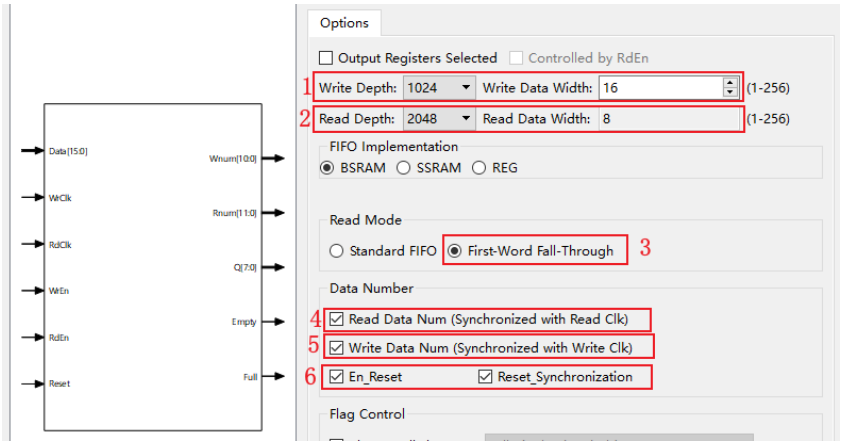

图 1-11 fifo_tx 配置界面

1.3.7 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。

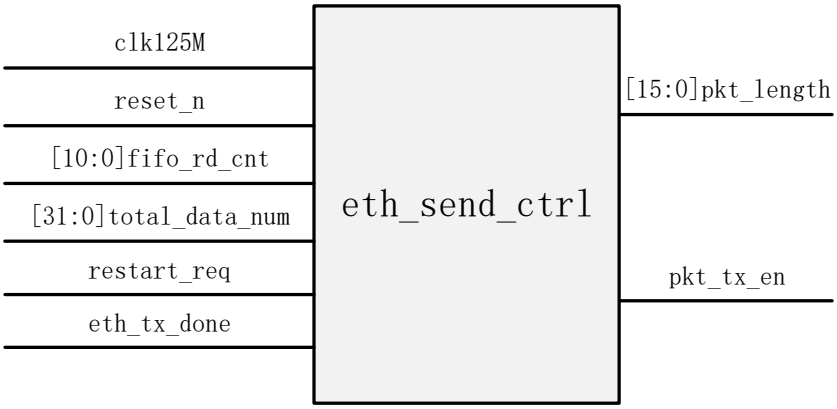

图 1-12 网口发送控制模块

对该模块的信号说明如下表 1-9 所示：

表 1-9 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
fifo_rd_cnt [10:0]	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
eth_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 fifo_rd_cnt 的数值满足一帧数据帧发送采集数据长度时，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(fifo_rd_cnt >= (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 eth_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(eth_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 ACG525 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，还需要给 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。

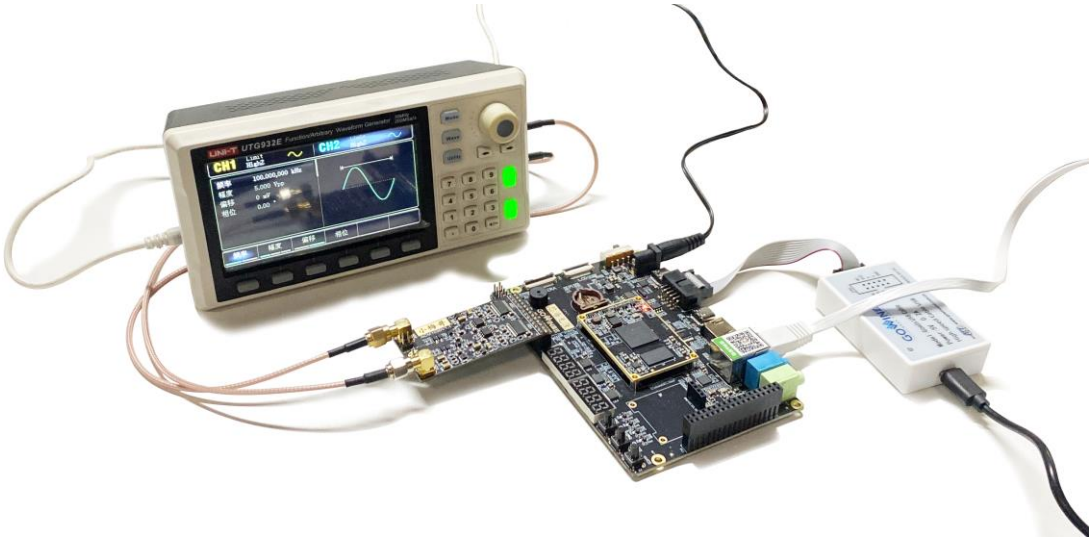

图 1-13 硬件连接图

连接完成之后，下载生成的数据流文件。文件下载成功之后，可以看到开发板上的 LED0 和 LED1 会被点亮，LED0 灯亮锁相环工作正常，LED1 灯亮 ddr3 初始化正常。

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。

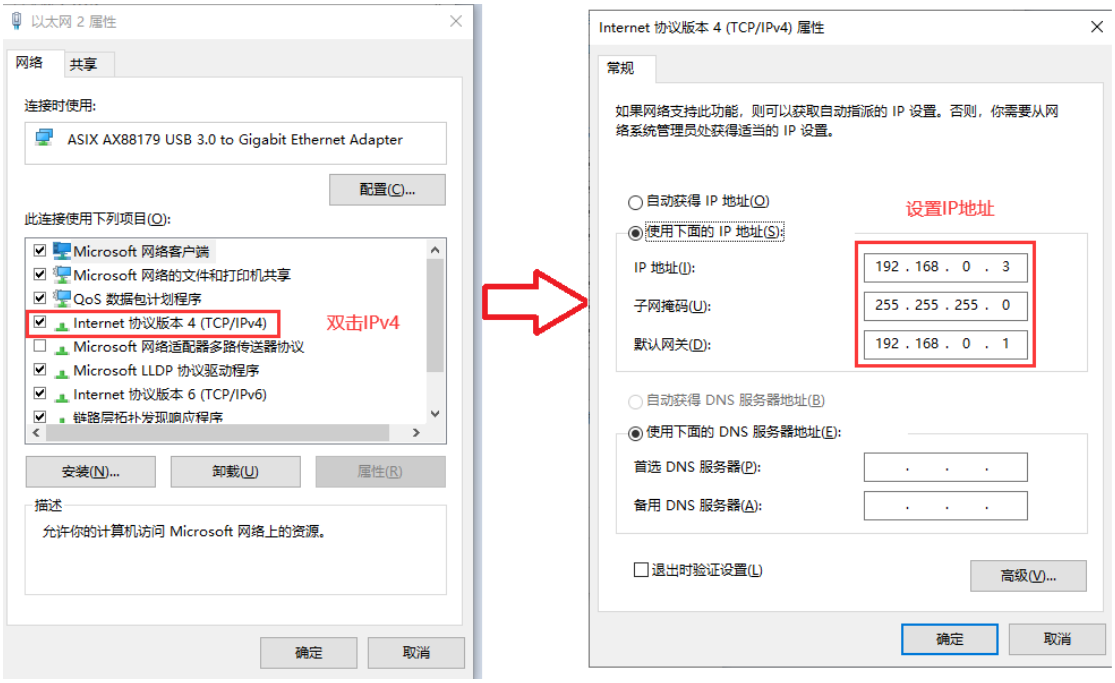

图 1-14 修改电脑 IP 地址

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内容：

以太网通信静态 ARP 绑定方法与常见问题解决方案

http://www.corecourse.cn/forum.php?mod=viewthread&tid=28645

(出处: 芯路恒电子技术论坛)

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。

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。

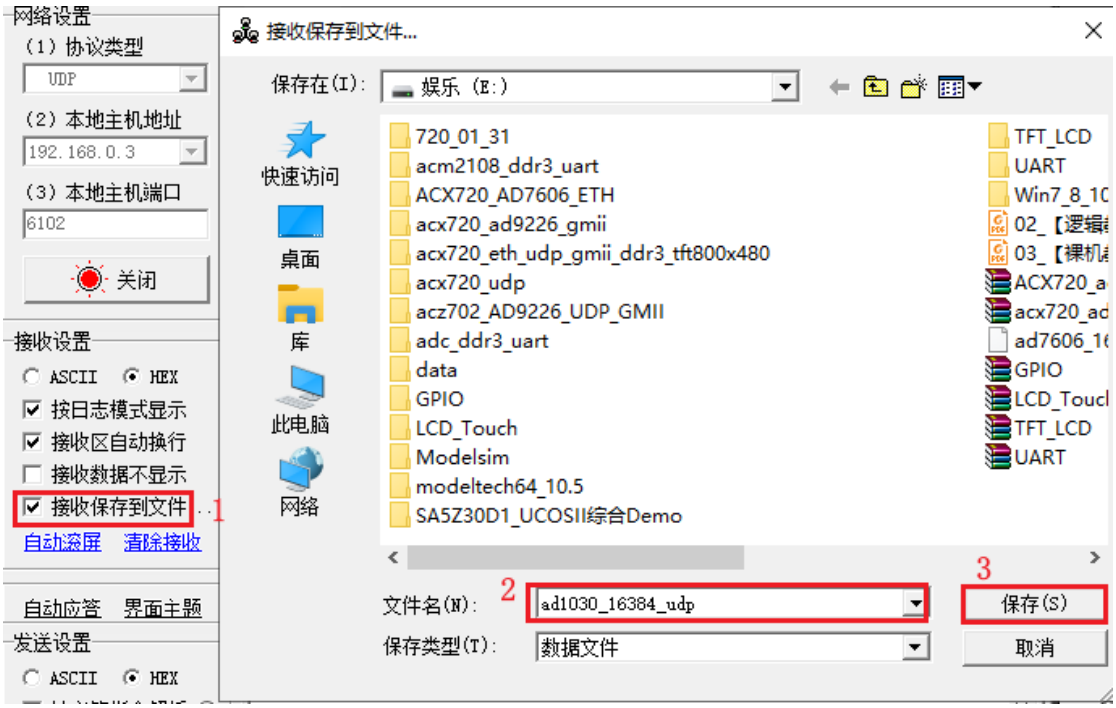

图 1-15 设置保存文件

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 1-10 所示：

表 1-10 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。

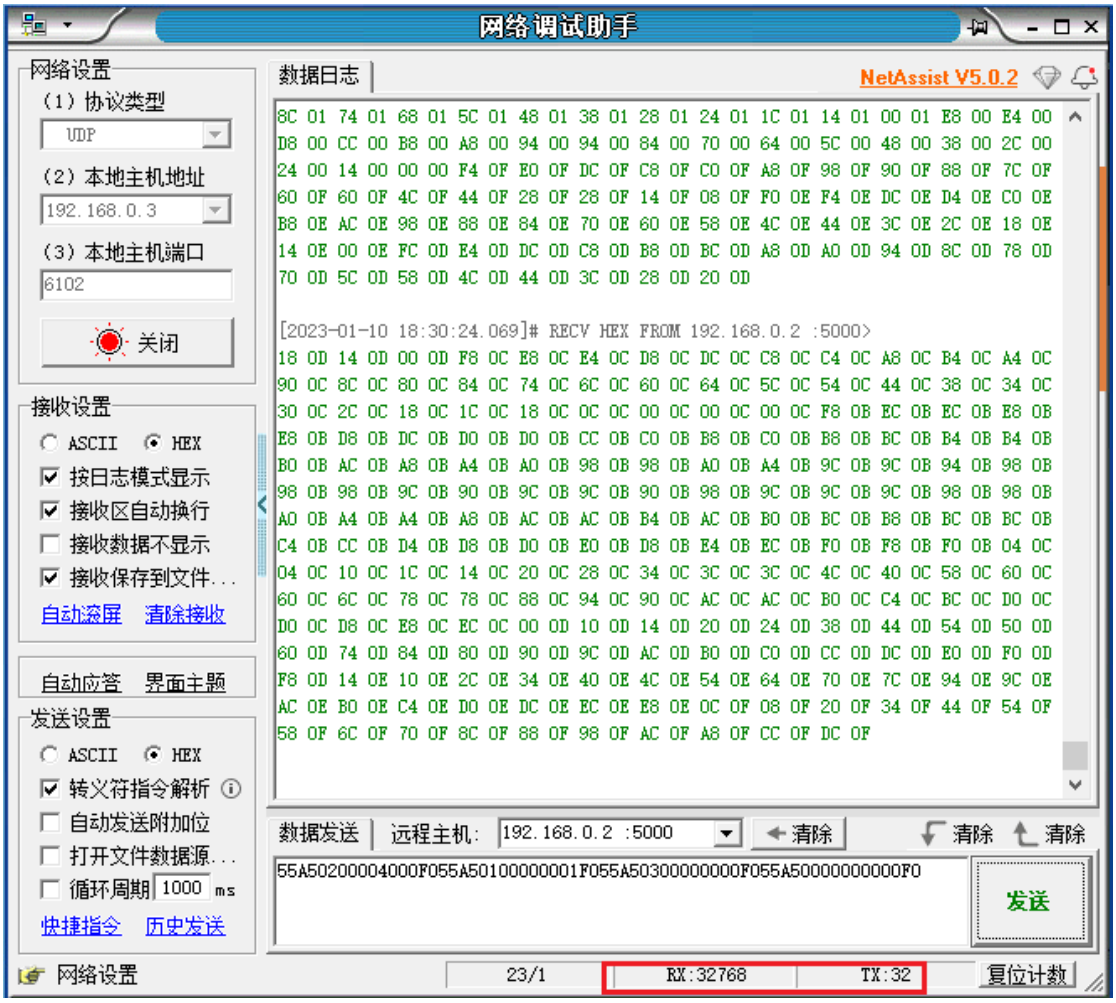

图 1-17 网络调试助手接收数据

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。

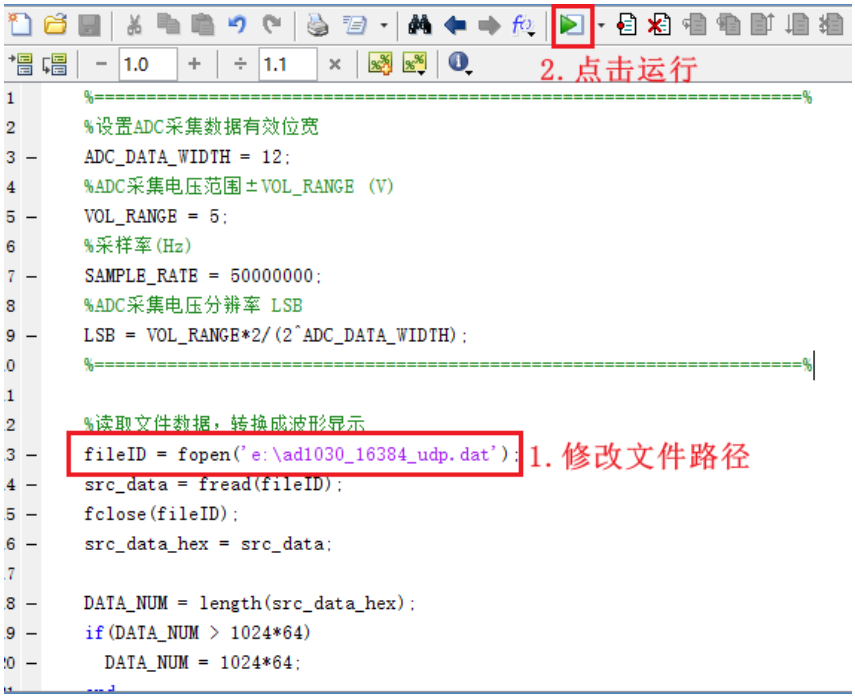

图 1-18 修改文件路径并运行

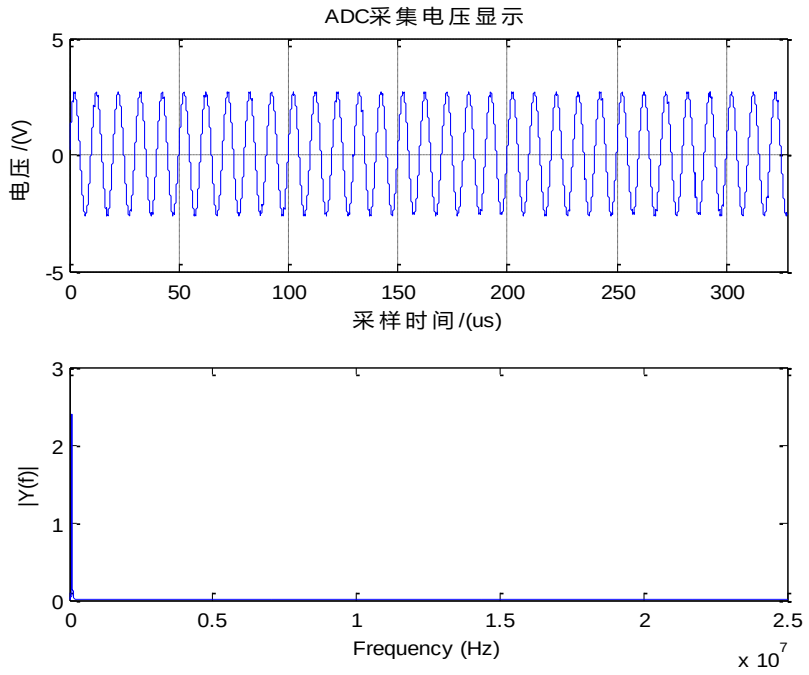

图 1-19 MATLAB 分析波形图

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，读者在我们提供的资料包中可以找到。双击上位机软件，初始界面如下图 1-20 所示。

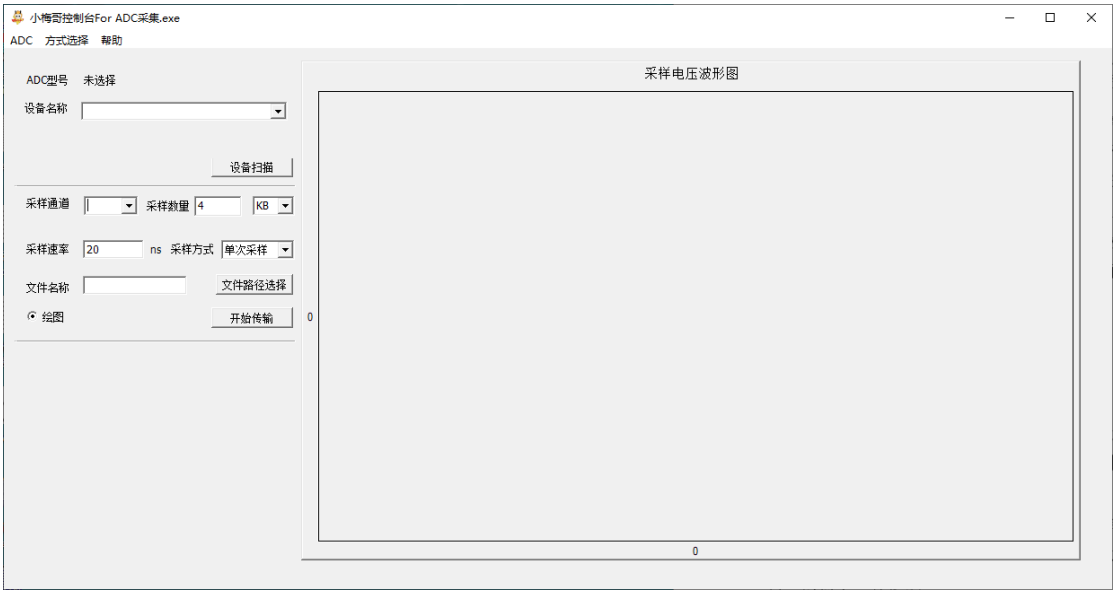

图 1-20 上位机软件初始界面显示

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。

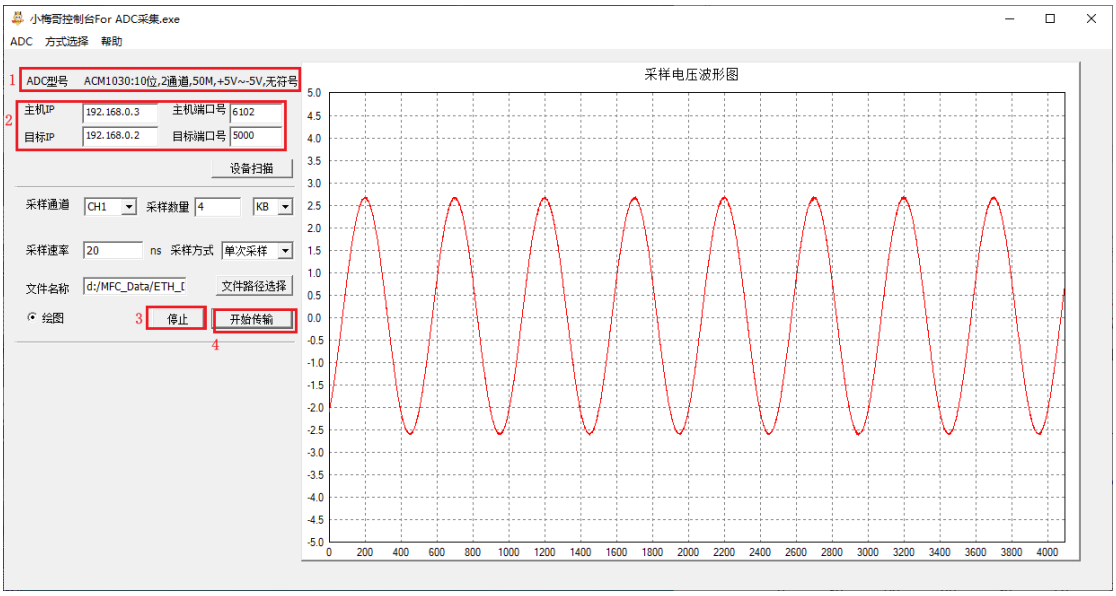

图 1-21 数据采集上位机显示图

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。

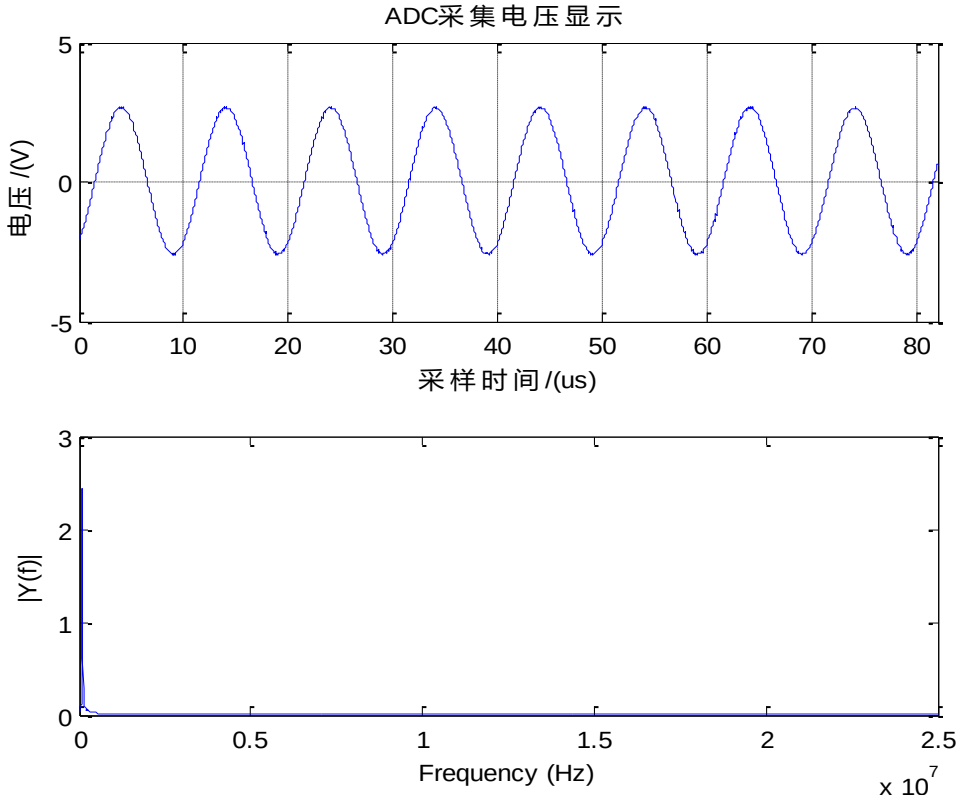

图 1-22 MATLAB 进一步波形分析图

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。