1 基于千兆以太网的 AD7606C 数据采集 DDR3 存储系统（串行驱动）

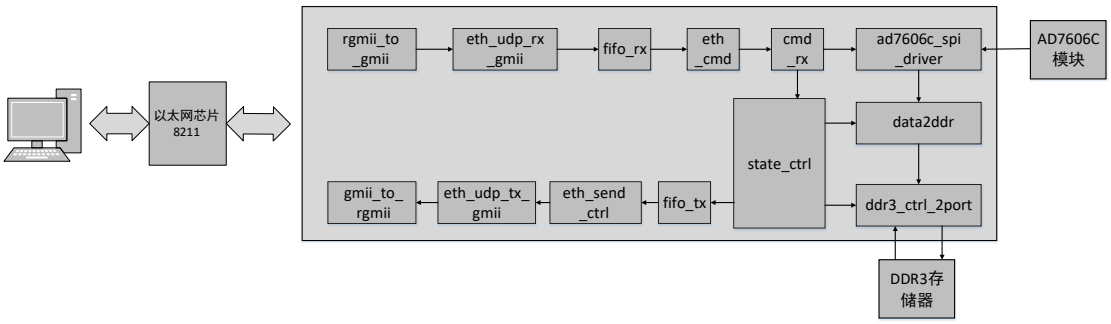
工程源码	|-ACX720 开发板 |-- ACX720_7606C_DDR3_SOFT_8DOUT_SPI_RGMII
	如果您手头的硬件不支持本实验，您可以学习本实验的理论内容，也可以跳过本节内容，继续后续内容的学习。

章节导读

本节实验结合 ADI 公司的 16 位 8 通道并行采样 ADC 芯片 AD7606C，并利用 ACX720 开发板上一片 Realtek 的 RTL8211 以太网收发器，实现了对 AD7606C 型 8 通道 16 位 ADC 的数据转换控制并输出。FPGA 采用 RGMII 接口与以太网 PHY 芯片通信，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 AD7606C 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 ACX720 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 AD7606C 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，AD7606C 开始采集数据，将 AD7606C 采集的数据存储至 DDR3 中，在配置寄存器的过程中要考虑到 DDR3 的写深度，一次采样所能采集的数据应该小于或等于 DDR3 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所示。



1-1 千兆以太网传输 AD7606C 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、mmcm 模块：锁相环模块，将 rgmii 接口时钟信号 rgmii_rx_clk_i 偏移 90°得到 rgmii_rx_clk 时钟信号，这样做是为了在时钟信号的上升沿/下降沿取数据时，取得数据刚好是数据信号 rgmii_rxd 的正中间，使得采样的数据处于最稳定的状态，如下图 1-2 所示。

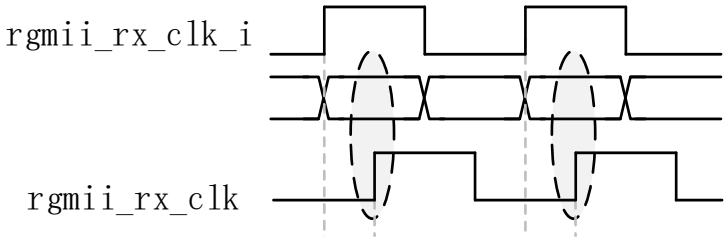


图 1-2 采集数据波形图

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，AD7606C 控制器工作时钟 100M，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、ad7606c_spi_driver 模块：AD7606C 控制器串行驱动模块，该控制器实现了对 AD7606C 型 8 通道 16 位 ADC 的数据转换控制并输出。使用该控制器时，用户无需关心 AD7606C 的具体控制时序，一切都在控制器内部完成，用户只需

要像使用并行 ADC 一样取用数据即可。

8、data2ddr 模块：ADC 采集控制模块，AD7606C 在配置好采样频率后进行采样，采集的八通道数据由 ad7606c_spi_driver 输入该模块。

9、ddr3_ctrl_2port 模块：DDR3 双端口模块，用来缓存 AD7606C 采集到的数据。

10、state_ctrl 模块：产生 DDR3 双端口模块所需要的控制信号以及产生写进 FIFO 的控制信号。

11、fifo_tx 模块：FIFO IP 核，存储从 ADC 采集控制模块输出的 16 位数据，数据经缓存后又由以太网发送模块读取。

12、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。

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。

这里不对 IP 核、rgmii_to_gmii 模块、eth_udp_rx_gmii 模块、ddr3_ctrl_2port 模块、eth_udp_tx_gmii 模块、gmii_to_rgmii 模块进行讲解，因为在提供的教学文档的相对应的章节有做讲解。还需要设计的模块包括eth_cmd 模块、cmd_rx 模块、data2ddr 模块、ad7606c_spi_driver 模块、state_ctrl 模块和 eth_send_ctrl 模块。

1.2 ACM7606C 模块简介

ACM7606C 数据采集模块使用的是 ADI 公司的 16 位 8 通道同步采样模数转换器 AD7606C，模块图如下图 1-3 所示。

图 1-3 AD7606C 模块图

AD7606C 是一款具有八通道的 16 位同时采样模拟到数字数据采集系统（DAS）。每个通道包含模拟输入箝位保护、可编程增益放大器（PGA）、低通滤波器（LPF）和 16 位逐次逼近寄存器（SAR）模数转换器（ADC）。还包含灵活的数字滤波器、低漂移 2.5 V 精密参考电压源、驱动 ADC 的参考缓冲器以及灵活的并行和串行接口。

同时所有通道均能以高达 1MSPS 的吞吐速率采样。输入箝位保护电路可以耐受最高达±21V 的电压。无论以何种采样频率工作，其模拟输入阻抗均为 1MΩ。这是一个固定的输入阻抗，不随 AD7606C 采样频率而变化。这种高模拟输入阻抗消除了在 AD7606C 前面的驱动器放大器的需要，允许直接连接到源或传感器。因此，双极电源可以从信号链上移除。

芯片对外提供 SPI 和并行的数字接口。当 AD7606C 的 8 个通道全部以 1MSPS 的最高速率进行转换时，数据输出速率达到 128Mbps，需要使用高性能 MCU 的 SPI 外设才能勉强该速率要求。因此可以使用 16 位并口来进行数据的传输，提高数据传输速率。当 AD7606C 应用在 FPGA 系统中的时候，使用 SPI 串行接口和并行接口都能够轻松的满足数据传输的速率需求。当在 FPGA 系统上应用 AD7606C 时，可以通过在 FPGA 上设计 AD7606C 控制转换逻辑，将转换结果数据直接存储到片上的存储器如 FIFO 或者 RAM 中，也可以存储到 FPGA 片外的存储器如 SRAM 或 SDRAM 中，然后由其他主控芯片如 MCU 或 DSP 读出，或者直接在 FPGA 内部进行数据的运算和处理。当然，由于 FPGA 片上可以设计软核控制器，也可以直接使用软核控制器完成数据的处理和传输工作。

1.2.1 功能框图

AD7606C 的功能框图如下图 1-4 所示。

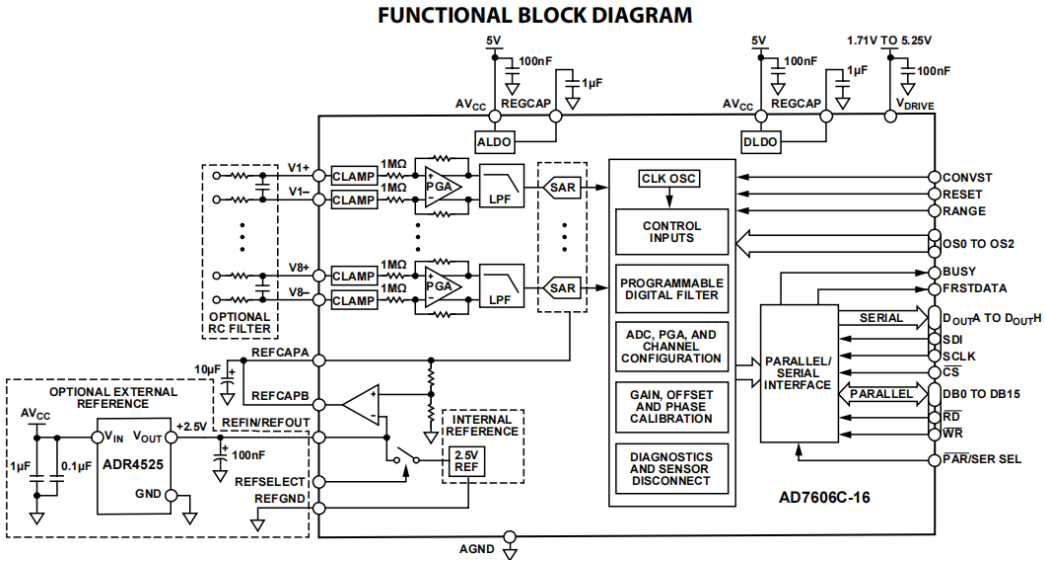


图 1-4 AD7606C 功能框图

如上图所示，当数据以串行模式输出时，数据会从 DoutA~DoutH 中输出，到底使用几根 DOUT 线进行输出，取决于寄存器的配置，数据可以通过 1 根、2 根、4 根、8 根 DOUT 线输出，如下图所示。

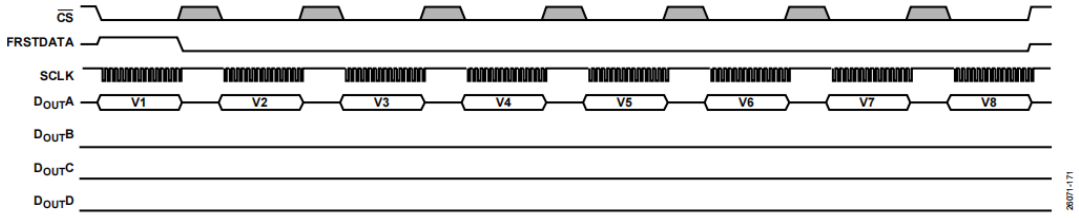


Figure 107. Serial Interface ADC Reading, One DOUTx Line

图 1-5 通过 1 根 DOUT 线进行输出

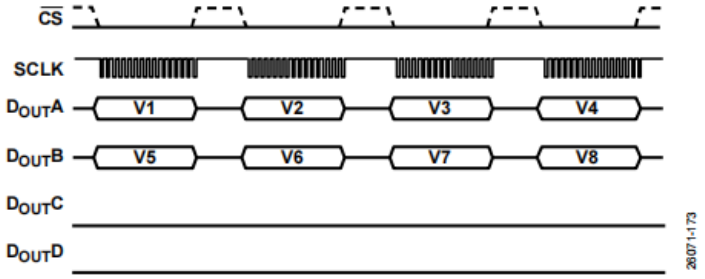


Figure 104. Serial Interface ADC Reading, Two DOUTx Lines

图 1-6 通过 2 根 DOUT 线进行输出

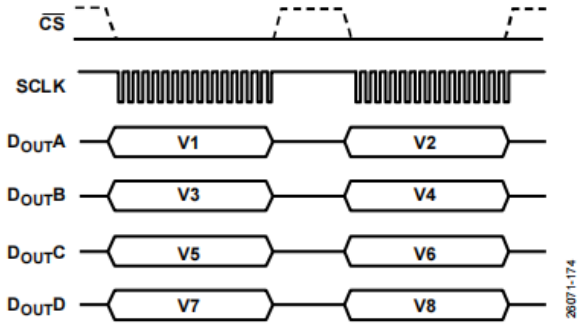


Figure 105. Serial Interface ADC Reading, Four DOUTx Lines

图 1-7 通过 4 根 DOUT 线进行输出

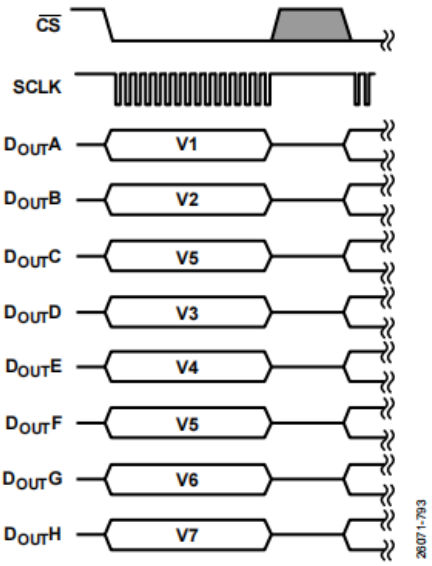


Figure 106. Serial Interface ADC Reading, Eight DOUTx Lines

图 1-8 通过 8 根 DOUT 线进行输出

如果数据是以并行方式输出，那么数据将从 DB[15:0]中输出。

1.2.2 模拟输入

AD7606C 可处理双极单端、单极单端。RANGE 引脚的逻辑电平决定所有模拟输入通道的模拟输入范围。如果此引脚与逻辑高电平相连，则所有通道的模拟输入范围为±10V。如果此引脚与逻辑低电平相连，则所有通道的模拟输入范围为±5V。AD7606C 的模拟输入阻抗为 1MΩ。这是固定输入阻抗，不随 AD7606C 采样频率而变化。这种高模拟输入阻抗消除了在 AD7606C 前面的驱动器放大器的需要，允许直接连接到源或传感器。因此，双极电源可以从信号链上移除。AD7606C 的输入结构如下图 1-9 所示，其各路模拟输入均含有箝位保护电路。虽然采用 5V 单电源供电，但此模拟输入箝位保护允许输入过压达到±21V。

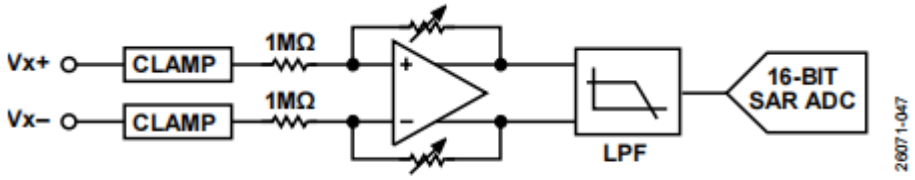


Figure 74. Analog Input Circuitry for Each Channel

图 1-9 AD7606C 模拟输入结构

1.2.3 数字滤波器与过采样

AD7606C 包含一个可选的数字平均滤波器，可以在需要更高信噪比或动态范围的较慢的吞吐量速率应用程序中启用。在硬件模式下，使用过采样引脚 OSx 来控制数字滤波器的过采样比，如下表 1-1 所示。OSx 引脚被锁定在 BUSY 信号的下降边缘或完全返回上。

表 1-1 可选引脚解码

OS2	OS1	OS0	过采样比
0	0	0	无过采样
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	进入软件模式

在软件模式下，如果所有 OSx 引脚都与逻辑高，则通过过采样寄存器（地址 0x08）选择过采样比。在软件模式下，还有两个额外的过采样比（128 过采样和 256 过采样）。

在过采样模式下，ADC 对 CONVST 信号的上升边缘上的每个通道进行第一个采样。转换第一个样本后，由内部生成的采样信号采集后续样本，如图 1-10 所示。

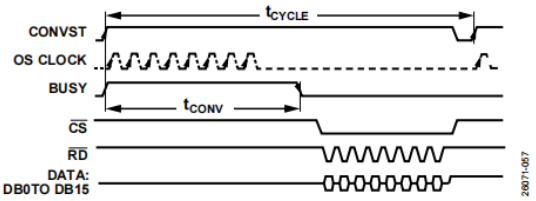


Figure 87. AD7606C-16 Oversampling by 8 Example, Read After Conversion, Parallel Interface, OS_CLOCK Is the Internally Generated Sampling Signal

图 1-10 AD7606C 过采样 8 为例

例如，如果配置了 8 个过采样，则取 8 个样本进行平均，并在输出上提供结果。CONVST 信号上升边缘触发第一个样本，其余 7 个样本用内部生成的采

样信号（OS_CLOCK）采集。因此，开启多个样本的平均功能会以降低最大吞吐量为代价来提高信噪比性能。当过采样函数打开时，BUSY 信号高时间（tCONV）会扩展。

当打开过采样时，转换时间（tCONV）会延长。必须降低吞吐量速率（1/tCYCLE），以适应更长的转换时间，并允许读取操作发生。为了在打开过采样时实现尽可能快的最快吞吐量速率，读取可以在 BUSY 信号高时间内执行读取，如在转换期间的读取部分中所述。

下表 1-2 和表 1-3 提供了用来选择不同的带宽模式以及不同的过采样倍率的过采样位解码。

表 1-2 低带宽不同过采样倍率的过采样位解码

OS [2:0]	过采样倍率	10V 范围 SNR(dB)	10V 范围 3dB 带宽 (kHz)	20V 范围 SNR(dB)	20V 范围 3dB 带宽 (kHz)	0V 到 10V 范围 SNR(dB)	0V 到 10V 范围 3dB 带宽(kHz)	最大吞吐量 CONVST 频率(kSPS)
000	No OS	91.5	25	92	25	89	25	1000
001	2	92.5	24.6	93	24.4	90	24.6	500
010	4	94.5	24	95	23.7	91	24	250
011	8	95	22.3	96	22.2	91.7	22.3	125
100	16	96	17.8	96.5	17.6	92.5	17.8	62.5
101	32	96.5	11.6	97	11.5	93	11.6	31.25
110	64	97	6.5	97.5	6.4	94.5	6.4	15.6

表 1-3 高带宽不同过采样倍率的过采样位解码

OS [2:0]	过采样倍率	10V 范围 SNR(dB)	10V 范围 3dB 带宽 (kHz)	20V 范围 SNR(dB)	20V 范围 3dB 带宽 (kHz)	0V 到 10V 范围 SNR(dB)	0V 到 10V 范围 3dB 带宽(kHz)	最大吞吐量 CONVST 频率(kSPS)
000	No OS	86	220	88	220	81	220	1000
001	2	89	154	91	154	84	155	500
010	4	91	97.5	93	97.5	86.5	97.5	250
011	8	92.5	53	94.5	53	88.5	53.5	125
100	16	95	27.5	96	27.5	90.5	27.5	62.5
101	32	96	13.8	97	13.7	92	13.5	31.25
110	64	97	7	97.5	7	93.5	7	15.6

1.2.4 工作时序

AD7606C 根据采样方式不同具有多种驱动时序，本次实验采用的为串行输出（使用 8 根 DOUT 线同时输出 8 个通道的数据），所以先要进行寄存器的配置。时序图由三部分组成：写寄存器时序、完成 AD 转换和读取 AD 数据。时序图如下所示。

Universal Timing Diagram

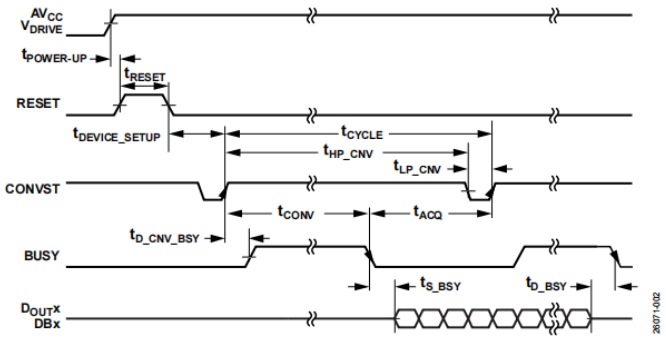


Figure 2. Universal Timing Diagram

图 1-11 通用时序

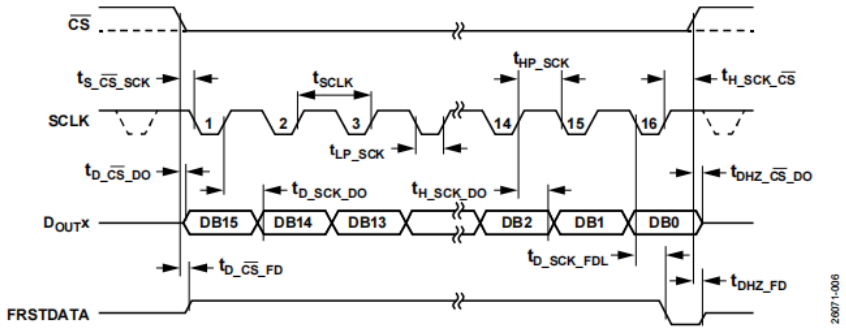


Figure 6. Serial Timing Diagram, ADC Mode (Channel 1)

图 1-12 通道 1 串行时序

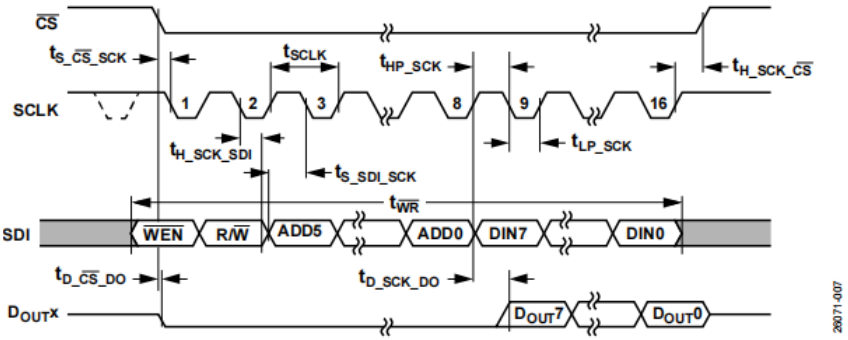


Figure 7. Serial Timing Interface, Register Map Read and Write Operations

图 1-13 寄存器串行模式

当 CONVST 变为上升沿时，BUSY 信号转变为高电平，代表转换开始，知道 BUSY 的下降沿到来，代表数据已经转换完成，正在锁存至输出数据寄存器中，当$\overline{CS}$变为下降沿时，数据将会被输送到总线上。当 V1 转换结果开始输出之后，FRSTDATA 会随后转变为高电平，表示输出数据总线可以提供 V1 的结果。

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 1-14 所示。

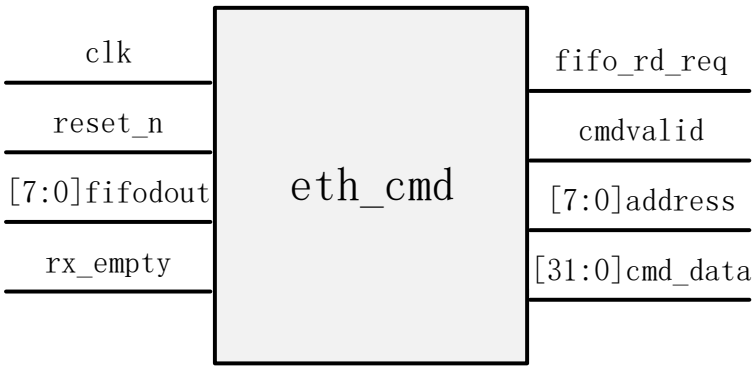


图 1-14 接收转命令模块框图

模块信号说明如下表 1-4 所示。

表 1-4 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 AD7606C 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 40 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 1-5 所示：

表 1-5 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 AD7606C 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
    end
    else
        cmdvalid <= #1 0;
```

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表 1-6 所示。

表 1-6 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	8	通道设置寄存器，共 8 位， 对应了 8 个通道的数据存储开关，如果某通道对应的设置为 1，则该通道的采样结果就会被存入 FIFO 并通过串口发送，注意对应的 2 进制的位，不是 10 进制，比如设置通道 3，对应 01 地址需要写入的值为 04(100)，而不是 03(011)。
DataNum	2	32	数据个数寄存器，设定总共采集传输多少个数据。注意，该寄存器设置的是总共采集的数据个数， 假设设置采集 100 个数据，ChannelSel 为 0000_0011b， 则实际每个通道采样的次数就是 50， 2 个通道的数据加起来是 100 个。假设设置采集 100 个数据，而且设置了 ChannelSel 为 0011_0011b，则实际每个通道采样的次数就是 25，4 个通道加起来采集 100 个数据
ADC_Speed_Set	3	32	ADC 采样速率设置寄存器。该寄存器用来设置 ADC 每多久执行一次转换。 由于 ADC 的最大采样速率为 1Msps，所以可以通过设置该寄存器的值来让 ADC 的采样速率在 1~1Msps 范围内调整，以适应不同的应用场景。ADC_Speed_Set=1000000000/10/speed-1，其中 speed 就是实际要设置的采样速率。
ADC_SOFT_DATA	4	32	AD7606C 软件模式配置寄存器，该寄存器用来配置 AD7606C 在软件模式下每个寄存器的参数。

指令转控制模块的结构框图如下图 1-15 所示。

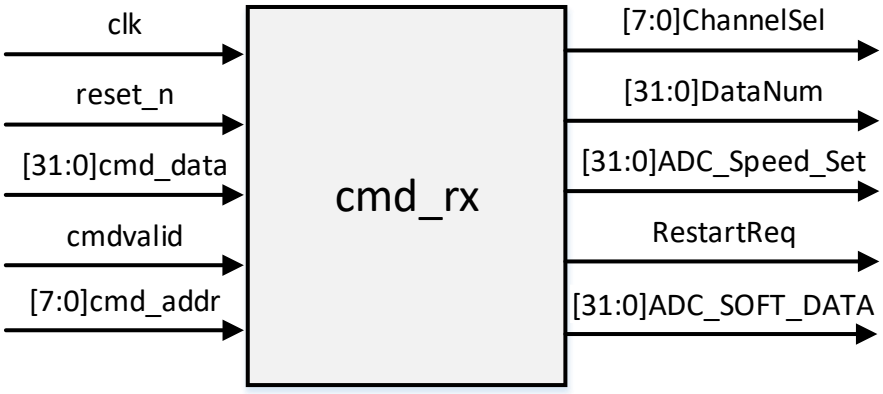


图 1-15 指令转控制模块结构框图

模块信号说明如下表 1-7 所示。

表 1-7 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[7:0]	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
ADC_SOFT_DATA[31:0]	O	软件配置寄存器

根据表 1-6 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[7:0]；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_addr 为 4 时，得到设置的寄存器配置的值，代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)begin
    ChannelSel <= 8'b1111_1111;
    DataNum <= 16'd32;
    ADC_Speed_Set <= 32'd9999;
    RestartReq <= 1'b0;
   ADC_SOFT_DATA <= 32'd0;
end
else if(cmdvalid)begin
  case(cmd_addr)
    0: RestartReq <= 1'b1;
    1: ChannelSel <= cmd_data[7:0];
    2: DataNum <= cmd_data[31:0];
```

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

1.3.3 AD7606C 控制器驱动模块

AD7606C 控制器驱动模块 ad7606c_spi_driver，该控制器实现了对 AD7606C 型 8 通道 16 位 ADC 的数据转换控制并输出。使用该控制器时，用户无需关心 AD7606C 的具体控制时序，一切都在控制器内部完成，用户只需要像使用并行 ADC 一样取用数据即可。该模块的结构框图如下图 1-16 所示。

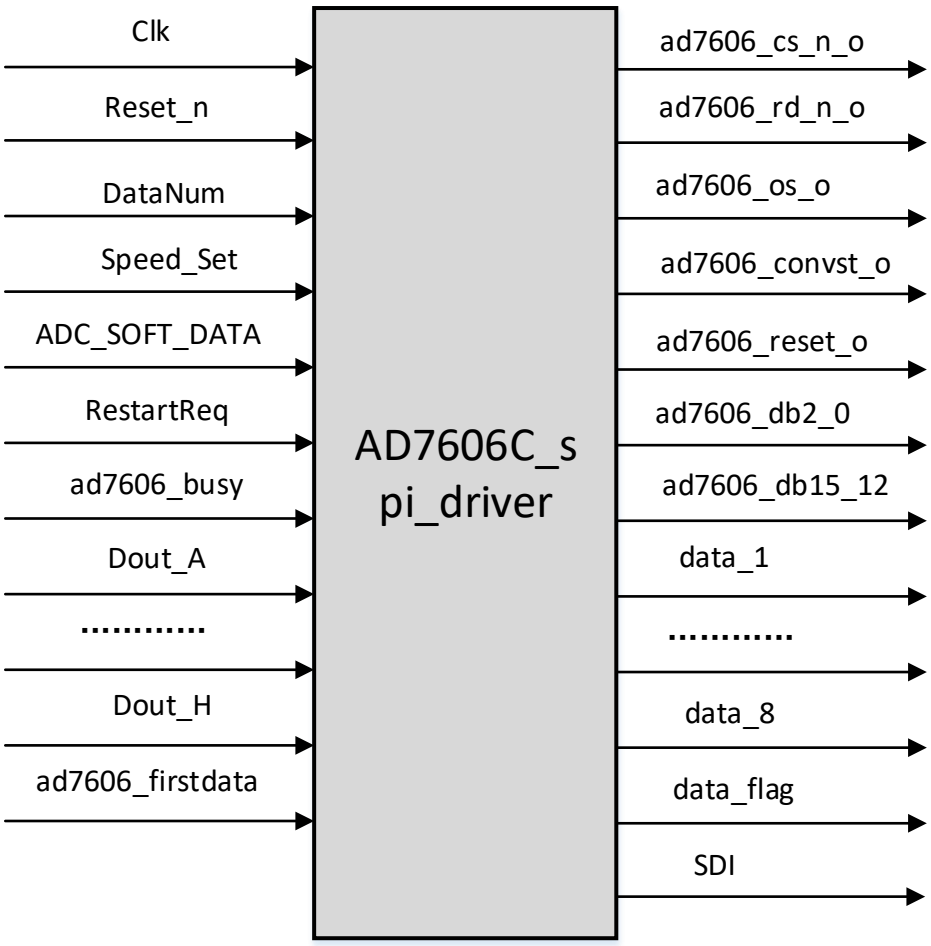


图 1-16 AD7606C 控制器串行驱动模块结构框图

该控制器接口分为四个类，第一类为时序逻辑模工作所必须的时钟和复位信号（Clk、Reset_n），第二类是 AD7606C 控制器与 AD7606C 芯片管脚相连的各种功能控制和数据信号，第三类是设置控制器工作状态/工作数据的用户控制

接口，第四类是控制器的结果输出接口。对于用户来说，只需要关注第三类和第四类接口的使用。对于该模块的信号说明如下所示。

表 1-8 AD7606C 控制器模块信号说明表

类	信号名称	I/O	信号意义
系统信号	Clk	I	模块系统时钟，这里使用 100M 时钟
	Reset_n	I	模块复位信号，低电平复位
控制信号	DataNum	I	采样数量控制端口
	Speed_Set	I	采样速率控制端口
	ADC_SOFT_DATA	I	寄存器配置端口
	convst_done	O	一次采样完成标志信号，单时钟周期脉冲信号。每 8 个通道结果都输出后，产生一个高脉冲信号
数据结果输出端口和标志信号	data_flag[7:0]	O	转换结果有效标志信号，由于使用 8 根 dout 线同时输出 8 个通道的数据，所以设置 8 个 Flag 信号，每个通道对应 Flag 信号的对应位
	data1[15:0]…data8[15:0]	O	8 个通道的采样结果输出端口，每个端口分别对应一个模拟通道的采样结果，使用时与 data_flag 信号配合，每当 data_flag 中的某一位为 1 时，则对应的通道上的 16 位采样结果已经就绪，可以使用。
ADC 芯片控制信号	ad7606_busy	I	ad7606C 转换状态标志信号，为高电平则表明 ad7606C 当前仍处于转换状态，结果没有更新，如果此时读取，读取的结果就还是前一次的采样转换结果。需要待该信号变为低电平之后，再读取 ad7606C 中的数据
	Dout_A…Dout_H	I	使用串行接口时，DB7/DOUTA 引脚作为 DOUTA 功能，B8/DOUTB 引脚作为 DOUTA 功能，详细的可参考 7606C 手册
	ad7606_cs_n_o	O	ad7606C 芯片选中控制信号，可以从 AD7606C 中读取转换结果时，需要使该信号为低电平
	ad7606_rd_n_o	O	选择串行接口时的串行时钟输入（SCLK）
	ad7606_os_o[2:0]	O	过采样模式引脚。OS0 至 OS2 选择过采样比率或启用软件模式
	ad7606_reset_o	O	ad7606C 的复位信号，复位 ad7606C 内部各个功能单元的工作状态
	ad7606_convst_o	O	ad7606C 转换开始信号，该信号的上升沿启动 ad7606C 内部的采样转换逻辑开始新一轮的采样转换
	ad7606_db2_0	O	在使用串行接口时，DB0~DB2 不使用
	ad7606_db15_12	O	在使用串行接口时，DB15~DB12 不使用
	SDI	O	在软件模式下，通过 SDI 配置 7606C 相关的寄存器

该控制器在工作时会根据主机的指令对采样频率进行修改，当信号转换完成后便对 ADC 写控制器发出 data_flag 信号，控制器将转换完成对应通道的数据写进 FIFO 或者 RAM。

1.3.4 ADC 采集控制模块设计

ADC 采集控制模块（data2ddr）的结构框图如下所示。将采集到的数据通

过 data2ddr 模块写进 DDR3 中。

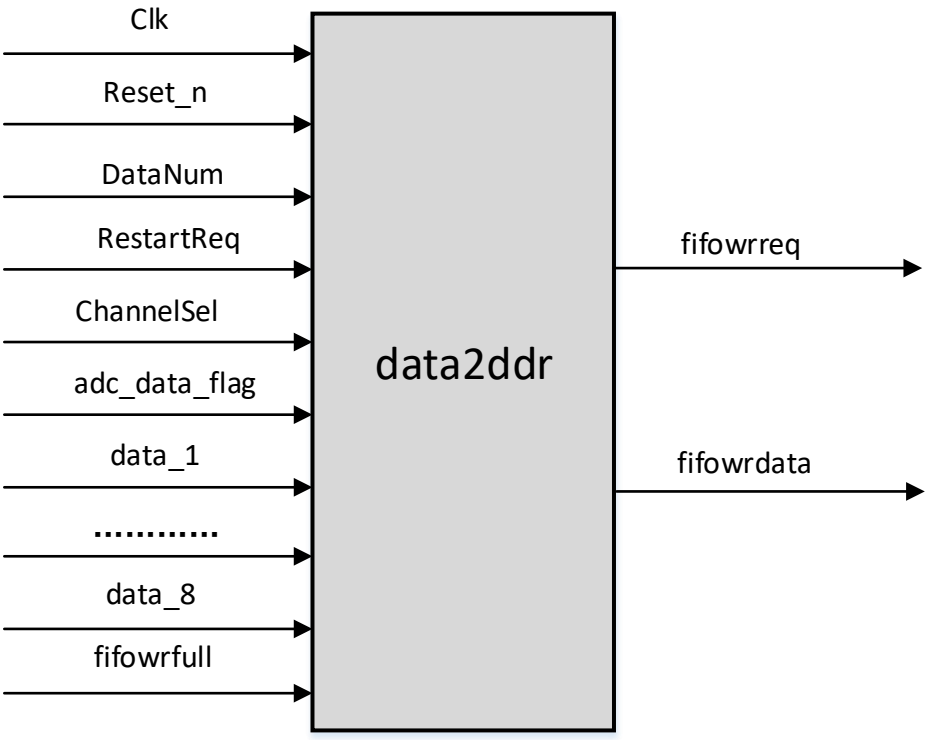


图 1-17 ADC 采集控制模块结构框图

对于该模块的信号说明如下表 1-9 所示。

表 1-9 ADC 采集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
DataNum [31:0]	I	单次采集的数据个数，最大不超过 FIFO 的深度
RestartReq	I	开始采样请求信号
ChannelSel [7:0]	I	需要采样的通道选择 ，每一位对应一个通道的开关
fifowrfull	I	FIFO 写满标志信号
adc_data_flag [7:0]	I	ADC 通道转换有效标志信号
data_1[15:0]…data_8[15:0]	I	数据输入端口
fifowrreq	O	采集到的数据写 FIFO 请求信号
fifowrdata[15:0]	O	采集到的写入到 FIFO 中的数据

接下来将对模块中的部分代码进行介绍。

首先，产生采样使能信号，每次采样信号到来开始采样，采样个数满了停止采样，代码如下所示：

```
    parameter SKIP_SAMPLES = 2;

    always@(posedge Clk or negedge Reset_n)
    if(!Reset_n)
```

```
      sample_en <= #1  1'b0;
    else if(RestartReq)
      sample_en <= #1  1'b1;
    else if(data_cnt >= DataNum + SKIP_SAMPLES)
      sample_en <= #1  1'b0;
```

这个 SKIP_SAMPLES 信号是为了舍弃前面几个不稳定的数据，也就是说在后面拉高 FIFO 写使能的时候，从第 SKIP_SAMPLES 个数据进行拉高。

随后对采样个数进行计数，AD7606C 在配置好采样频率后进行采样，采集的数据输入 ADC 采集控制模块。同时，输入 adc_data_flag 采样标志信号，使不同通道采集数据有效时产生对应位的有效信号。在采样使能阶段，每个 adc_data_flag 信号到来并且 ChannelSel 有效的时候计数器自加 1，代码如下所示：

```
    always@(posedge Clk or negedge Reset_n)
    if(!Reset_n)
        data_cnt <= #1  15'd0;
    else if(sample_en)begin
        if(adc_data_flag & ChannelSel)
            data_cnt <= #1  data_cnt + 1'd1;
        else
            data_cnt <= #1  data_cnt;
    end
    else
      data_cnt <= #1  15'd0;
```

接着设置延迟计数器，根据延迟计数器控制拉高的 FIFO 的写使能，代码如下所示：

```
    reg [1:0] skip_cnt;
    always @(posedge Clk or negedge Reset_n)
    if (!Reset_n)
        skip_cnt <= 2'd0;
    else if (RestartReq)
        skip_cnt <= 2'd0;
    else if (sample_en && (adc_data_flag & ChannelSel) &&
skip_cnt < SKIP_SAMPLES)
         skip_cnt <= skip_cnt + 1;
```

ADC 采集控制模块根据 adc_data_flag&ChannelSel 有效、sample_en 有效以及延迟计数器大于设定的值的情况下输出 fifowrreq 信号用于读取所需通道的数据，并将采集到的数据进行输出，通过通道索引值选择写入 FIFO 的数据，代码如下所示：

```
    always@(posedge Clk)
      fifowrreq <= #1  (adc_data_flag & ChannelSel)  &&
sample_en && skip_cnt >= SKIP_SAMPLES;
```

```
    always @(posedge Clk) begin
        case (channel_index)
            3'd0: fifowrdata <= data_1;
            3'd1: fifowrdata <= data_2;
            3'd2: fifowrdata <= data_3;
            3'd3: fifowrdata <= data_4;
            3'd4: fifowrdata <= data_5;
            3'd5: fifowrdata <= data_6;
            3'd6: fifowrdata <= data_7;
            3'd7: fifowrdata <= data_8;
            default: fifowrdata <= 16'd0;
        endcase
    end
```

1.3.5 state_ctrl 控制模块

状态控制模块 state_ctrl 的作用就是控制 ADC 的采集，以及向 DDR 读写数据等相关状态的控制，状态控制模块（state_ctrl）的结构框图如下所示。

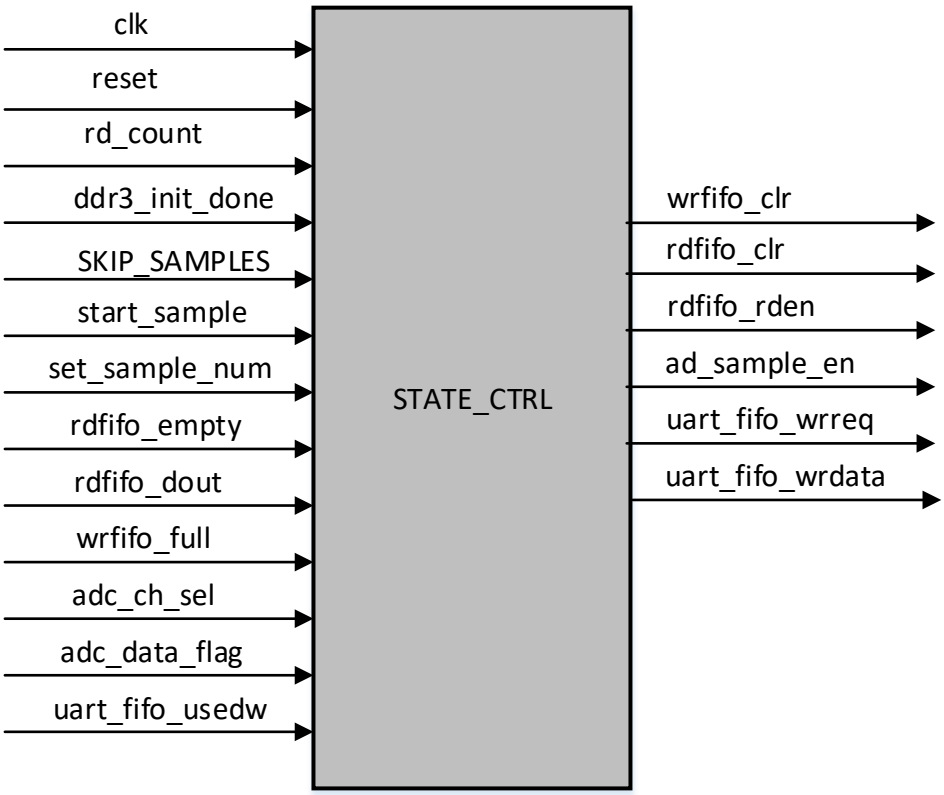


图 1-18 state_ctrl 模块接口框图

对于该模块的信号说明如下所示。

信号名称	I/O	信号意义

clk	I	系统时钟
reset	I	系统复位，高电平有效
rd_count	I	FIFO 的可读数量
ddr3_init_done	I	DDR3 初始化完成标志
SKIP_SAMPLES	I	舍弃前 SKIP_SAMPLES 个数据
start_sample	I	ADC 模块开始采样标志信号
set_sample_num	I	设置的采样个数
rdfifo_dout	I	读 FIFO 的读数据输出，数据位宽为 16 位
rdfifo_empty	I	读 FIFO 的读空标识信号，用于标识当前 FIFO 是否为空
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
adc_ch_sel	I	设置的 ADC 的采样通道
adc_data_flag	I	ADC 模块转换结果有效标志信号
eth_fifo_usedw	I	以太网发送数据流的模块的写 FIFO 计数
wrfifo_clr	O	FIFO 的写清除信号，wrfifo_clr 向外打三拍输出，保证 wrfifo 的清零信号的生效节拍数
rdfifo_clr	O	读 FIFO 清空控制信号，给高电平表示执行清空，执行清空操作时，需保证给 3 个及以上个时钟（rdfifo_clk）周期的高电平
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
ad_sample_en	O	ADC 采样使能标志信号
eth_fifo_wrreq	O	以太网 FIFO 的写请求信号
eth_fifo_wrdata	O	以太网 FIFO 需要发送的 16 位数据

下面我们将对每个状态的作用和具体的实现代码进行说明。

1、IDLE 状态

当处于空闲状态并且 DDR 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 && 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_FIFO_CLEAR 状态。空闲状态代码代码如下所示：

```
IDLE:
begin
    if(start_sample_rm)begin
        state<=DDR_WR_FIFO_CLEAR;
    end
    else begin
        state<=state;
    end
  end
```

2、DDR_WR_FIFO_CLEAR 状态

当进入写 FIFO 清零状态后，开始清除写 FIFO 内的原始数据。设置清除 DDR 写 FIFO 的计数器，保证至少 10 拍的延时，确保 FIFO 中的数据清除完毕，代码如下所示：

```
always@(posedge clk or posedge reset)begin
if (reset)
    wrfifo_clr_cnt<=0;
else if(state==DDR_WR_FIFO_CLEAR)
begin
    if(wrfifo_clr_cnt == 9)
        wrfifo_clr_cnt<= 9;
    else
        wrfifo_clr_cnt<= wrfifo_clr_cnt + 1'b1;
end
else
    wrfifo_clr_cnt<= 0;
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，代码如下。

```
DDR_WR_FIFO_CLEAR:
begin
    if(!wrfifo_full && (wrfifo_clr_cnt==9))
        state<= ADC_SAMPLE;
    else
        state<=state;
end
```

当处于 DDR_WR_FIFO_CLEAR 状态时，我们需要产生清除写 FIFO 的信号

wrfifo_clr，由三拍延时信号拉高提供，之所以提供的延迟的信号时间为 3 拍，是为了给清 FIFO 信号足够的拉高时间，以保证清空指令的可靠，也就是在 wrfifo_clr_cnt 为 0、1 或 2 时，wrfifo_clr 置 1，否则 wrfifo_clr 为 0。代码如下所示：

```
always@(posedge clk or posedge reset)begin
if (reset)
    wrfifo_clr<=0;
else if(init_done==1'b0)
    wrfifo_clr<=1'b1;
else if(state==DDR_WR_FIFO_CLEAR)
begin
    if(wrfifo_clr_cnt==0||wrfifo_clr_cnt==1||wrfifo_clr_cnt==2)
        wrfifo_clr<=1'b1;
    else
        wrfifo_clr<=1'b0;
end
else
    wrfifo_clr<=1'b0;
end
```

3、ADC_SAMPLE 状态

进入 ADC 采样数据状态之后， ADC 采样个数计数器 adc_sample_cnt 开始计数，代码如下所示：

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

当 adc_sample_cnt 达到设定的采样数据个数的时候，ADC 模块数据采集完成，跳转到读 FIFO 清除状态，代码如下所示：

```
ADC_SAMPLE:
begin
if((adc_sample_cnt>=set_sample_num + SKIP_SAMPLES))
    state<=DDR_RD_FIFO_CLEAR;
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_FIFO_CLEAR 状态

当进入读 FIFO 清零状态后，做和前面写 fifo 清零相同的操作，发出三拍的清零指令，同时保证一个 10 拍的基本延时，代码如下。

```
always@(posedge clk or posedge reset)begin
if (reset)
    rdfifo_clr_cnt<=0;
else if(state==DDR_RD_FIFO_CLEAR)
begin
    if(rdfifo_clr_cnt==9)
        rdfifo_clr_cnt<= 9;
    else
        rdfifo_clr_cnt<= rdfifo_clr_cnt + 1'b1;
end
else
    rdfifo_clr_cnt<= 0;
end
```

然后等待 rdfifo_empty 的信号拉低，拉低后，表示可以从 FIFO 里读出数据，此时进入下一个状态。

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

当处于 DDR_RD_FIFO_CLEAR 状态时，我们需要产生清除读 FIFO 的信号 rdfifo_clr，由三拍延时信号拉高提供，之所以提供的延迟的信号时间为 3 拍，是为了给清 FIFO 信号足够的拉高时间，以保证清空指令的可靠，也就是在 rdfifo_clr_cnt 为 0、1 或 2 时，rdfifo_clr 置 1，否则 rdfifo_clr 为 0。代码如下所示：

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

```
if (reset)
    rdfifo_clr<=0;
else if(init_done==1'b0)
    rdfifo_clr<=1'b1;
else if(state==DDR_RD_FIFO_CLEAR)
begin
    if(rdfifo_clr_cnt==0||rdfifo_clr_cnt==1||rdfifo_clr_cnt==2)
        rdfifo_clr<=1'b1;
    else
        rdfifo_clr<=1'b0;
end
else
    rdfifo_clr<=1'b0;
end
```

5、DATA_SEND_START 状态

进入到 DATA_SEND_START 状态的时候，状态机也进行跳转，代码如下。

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

6、DATA_SEND_WORKING 状态

进入数据发送状态之后，当发送数据计数器 send_data_cnt 计数到需要采集的数据个数 set_sample_num 时，跳转到 IDLE 状态，完成一次数据采集发送，当以太网发送中写入的数据小于 4096 的时候，并且读 FIFO 中可读数量大于 16 的时候，开始从 DDR 中读取出数据写入到 FIFO 中，当然这个数据并不是固定的，可以修改，代码如下所示：

```
DATA_SEND_WORKING: //6
begin
 //*//如果 send_data_cnt（USB 发送计数）等于给定的值,则跳转进入 IDLE 状态
    //如果整个数据块发送完成，则大循环收口
    if(send_data_cnt>=set_sample_num-1)begin
        state <= IDLE;
        rdfifo_rden <= 1'b0;
    end
    else if((eth_fifo_usedw < 4096)&& (rd_count > 16)) begin
        rdfifo_rden <= 1'b1;
        state <= DATA_SEND_WORKING;
    end
    else begin
    //*//每发送一个 16bit 数据，如果不满足 if 条件，则重新回到本状态
        rdfifo_rden <= 1'b0;
        state <= DATA_SEND_WORKING;
    end
```

```
end
```

当 rdfifo_rden 为 1 的时候，每个时钟上升沿到来之后，send_data_cnt 计数值加 1，实现对以太网发送的数据进行计数，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
```

当 rdfifo_rden 信号到来之后，我们需要产生串口写 FIFO 请求信号并且需要将从 DDR 读出的数据提取出来，代码如下所示：

```
always@(posedge clk or posedge reset)
if(reset) begin
   eth_fifo_wrreq <= 1'b0;
   eth_fifo_wrdata <= 16'd0;
end
else if(rdfifo_rden) begin
   eth_fifo_wrreq <= 1'b1;
   eth_fifo_wrdata <= rdfifo_dout;
end
else begin
   eth_fifo_wrreq <= 1'b0;
   eth_fifo_wrdata <=16'd0;
end
```

1.3.6 FIFO 数据存储模块

FIFO 模块需要接收从 ADC 采集控制模块输出的 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 100M，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 VIVADO 软件中点击 IP Catalog，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 1-19 所示。

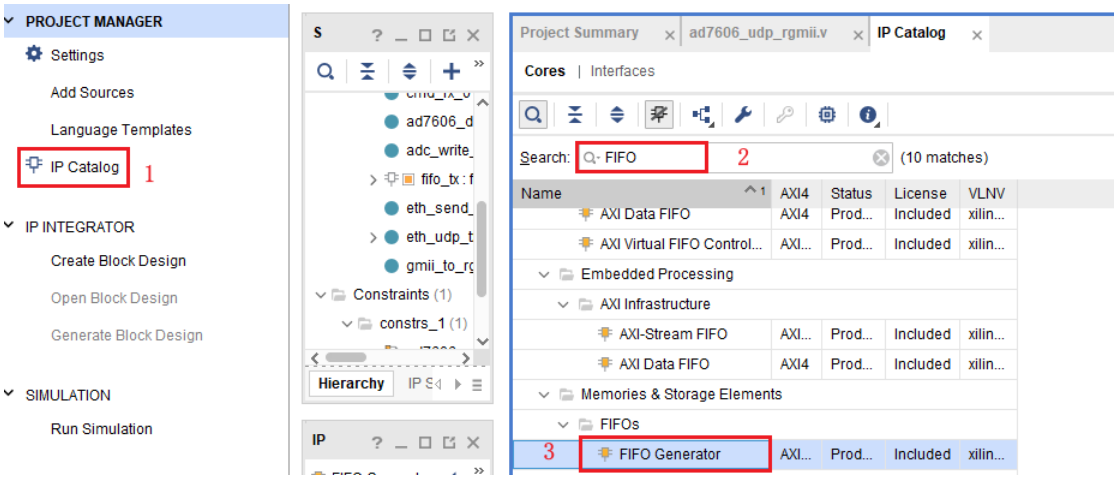


图 1-19 搜索 FIFO IP

双击 FIFO Generator 之后，进入 FIFO 配置界面。

首先将 FIFO 的名称修改为 fifo_tx，接口类型设置为 Native，FIFO 类型上根据使用的资源以及读写时钟是否相同分为多种，这里创建一个独立读写时钟，使用嵌入式 Block RAM 资源的 FIFO，选择 Independent Clocks Block RAM。操作如下图 1-20 所示。创建独立读写时钟是因为 ADC 驱动模块的工作时钟为 100M，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。

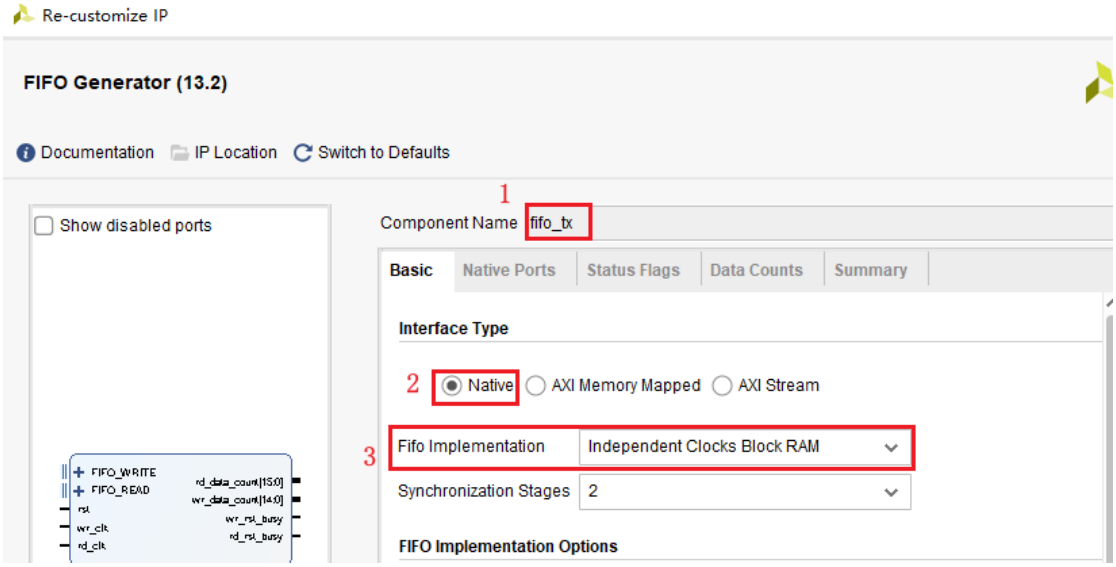


图 1-20 选择独立时钟 FIFO

然后点击“Native Ports”进行配置，配置如下：

1、读模式选择“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 dout 上。

2、写位宽设置为 16，写入深度设置为 32768。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，这里设置为 32768，那么代表单次采集的数据个数应该小于或等于 32768。

3、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。

配置界面如下图 1-21 所示。

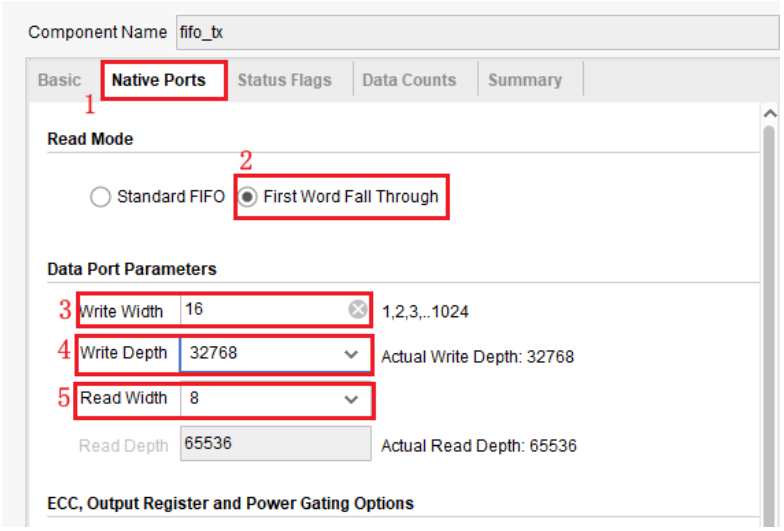


图 1-21 “Native Ports”配置界面

在“Data Counts”界面勾选“Write Data Count”和“Read Data Count”。FIFO 数据量计数信号输出， Write Data Count 和 Read Data Count 分别同步于写时钟和读时钟。位宽可以根据实际进行设置，这里保持默认位宽 15、16。配置界面如下图 1-22 所示。

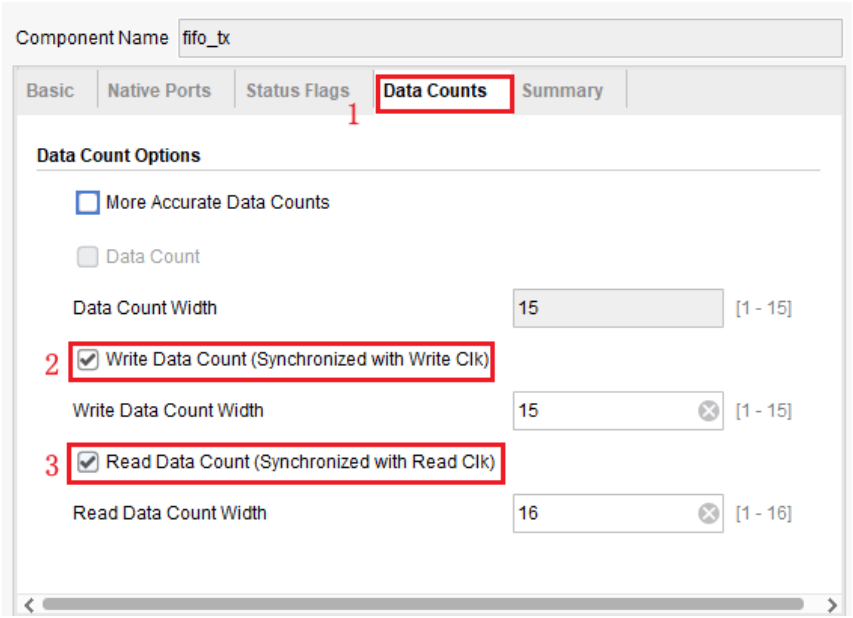


图 1-22 “Data Counts”配置界面

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。

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，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**错误!未找到引用源。**所示。

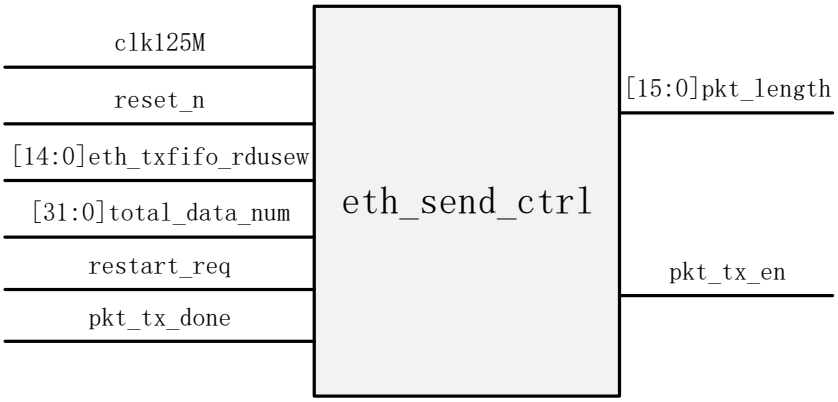


图 1-23 网口发送控制模块

对该模块的信号说明如下表 1-10 所示。

表 1-10 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，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

为保证以太网有效传输效率，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 AD7606C 采集的数据最大长度为 1472 字节。

在模块中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 ADC 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。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 ACX720 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，从而实现对 AD7606C 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，AD7606C 开始采集数据，将 AD7606C 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，然后通过上位机进行绘制。也可直接使用我们提供的上位机进行数据采集。

1.4.1 硬件连接

将 ACM7606C 模块、网线、下载器、电源线依次连接在开发板上，需要注意 ACM7606C 模块连接正确后右边会多出 6 组排针，由于视觉缘故，这里很容易连接错误，并且需要将 ACM7606C 模块上的 H2 接到串行接口上，如下图 1-24 所示。

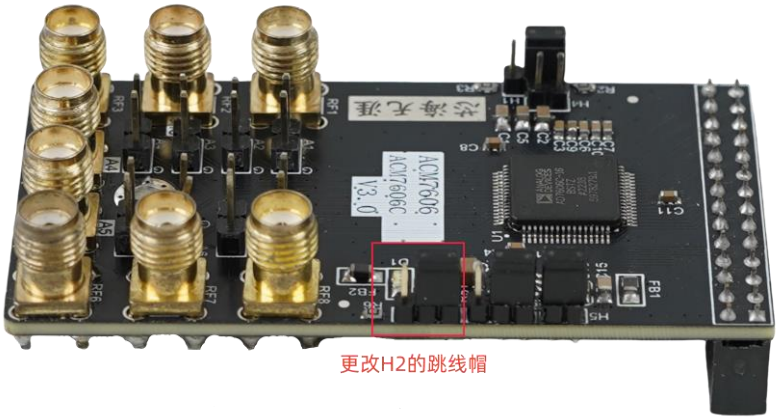


图 1-24 更改跳线帽

还需要给 AD7606C 连接信号源，这里我们连接的是 AD7606C 的通道 1，信号源给的是 1khz，vpp=5V 的正弦波，整体的硬件连接图如下所示。

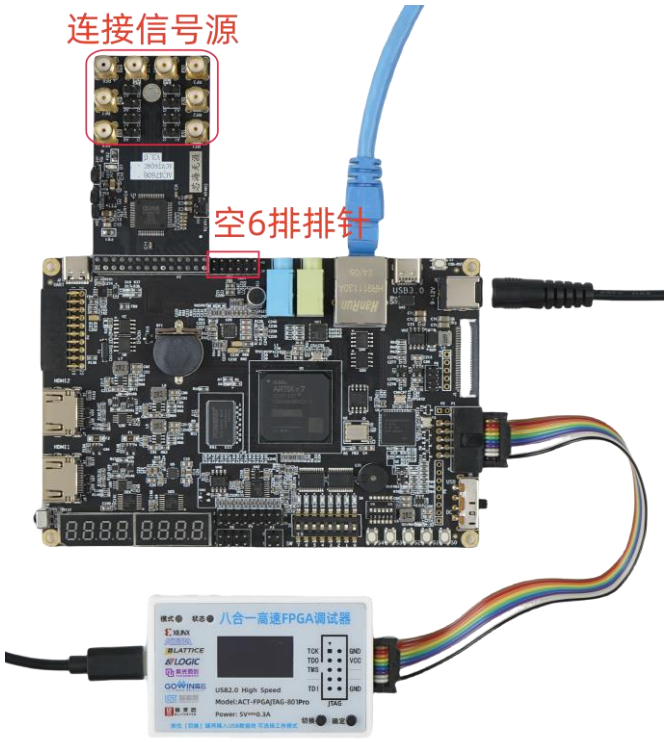


图 1-25 硬件连接图

1.4.2 下载 bit 文件

将生成的 bit 文件下载到开发板上，进行验证，下载方式如下所示。

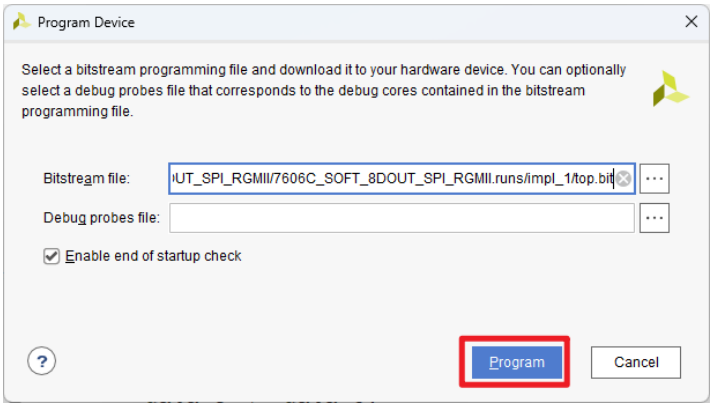


图 1-26 下载 bit

1.4.3 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 1-27 所示。

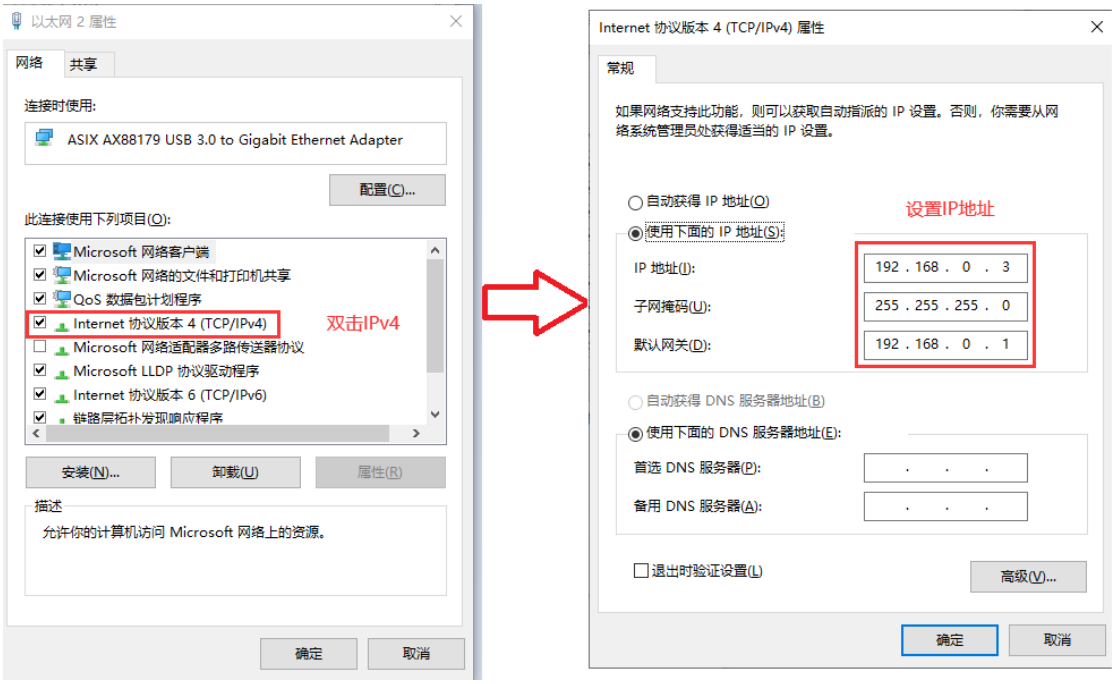


图 1-27 修改电脑 IP 地址

1.4.4 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 的绑定请查看以下帖子内容：

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)

1.4.5 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盘 A ACX720 开发板标准配套资料\05_常用软件中找到。

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、发送设置中数据类型设置为 hex 格式。

7、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.

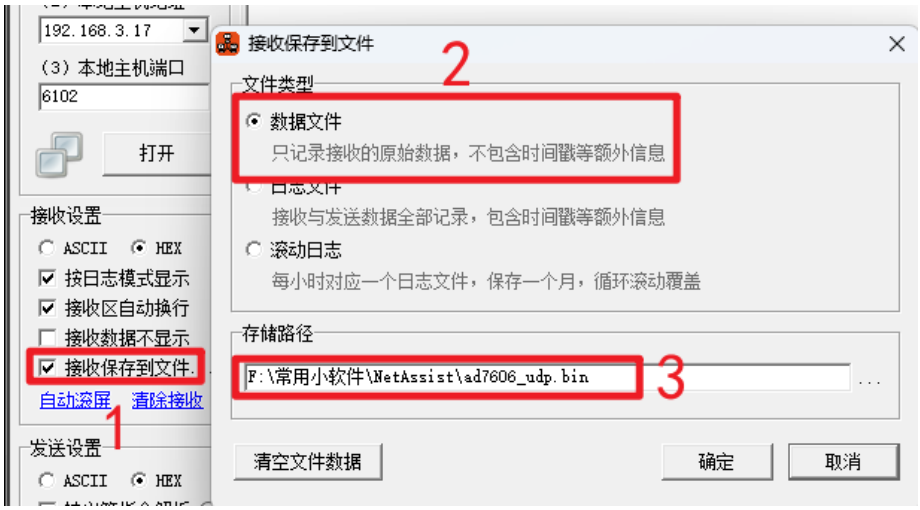


图 1-28 设置保存文件

在前面接收转命令模块中介绍到数据帧格式对 AD7606 的五个寄存器进行配置。

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 1M，则对应的 ADC_Speed_Set 值为 1000000000/10/1000000 - 1 = 99（0x00000063），则发送的数据帧内容为：0x55 0xA5 0x03 0x00 0x00 0x00 0x63 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，AD7606C 就能实现以 1M 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 32768 个数据。五个寄存器对应的配置如下表 1-11 **错误!未找到引用源。**所示：

表 1-11 AD7606C 数据帧格式配置表

寄存器名称	数据帧数据
DataNum	55 A5 02 00 00 80 00 F0
ChannelSel	55 A5 01 00 00 00 01 F0
ADC_Speed_Set	55 A5 03 00 00 00 63 F0
ADC_SOFT_DATA	55 A5 04 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：

55A50400000000F0 55A50200008000F0 55A50100000001F0 55A50300000063F0 55A50000000000F0

最终的网络助手配置界面如下图 1-29 所示。

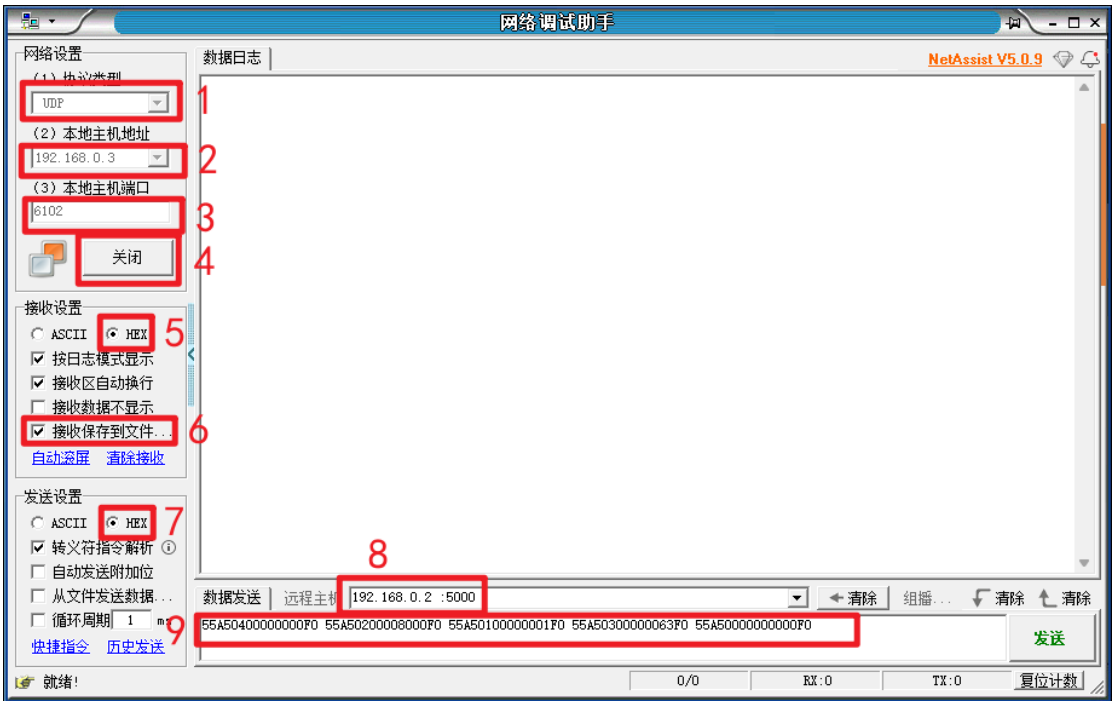


图 1-29 网络调试助手配置界面

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 1-30 所示。从图中可以看出一共接收到 65536 个数据，这是因为设置的 ADC 采样数量为 32768，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 32768*2 个数据，这也就是说明接收到的数据的个数没错。

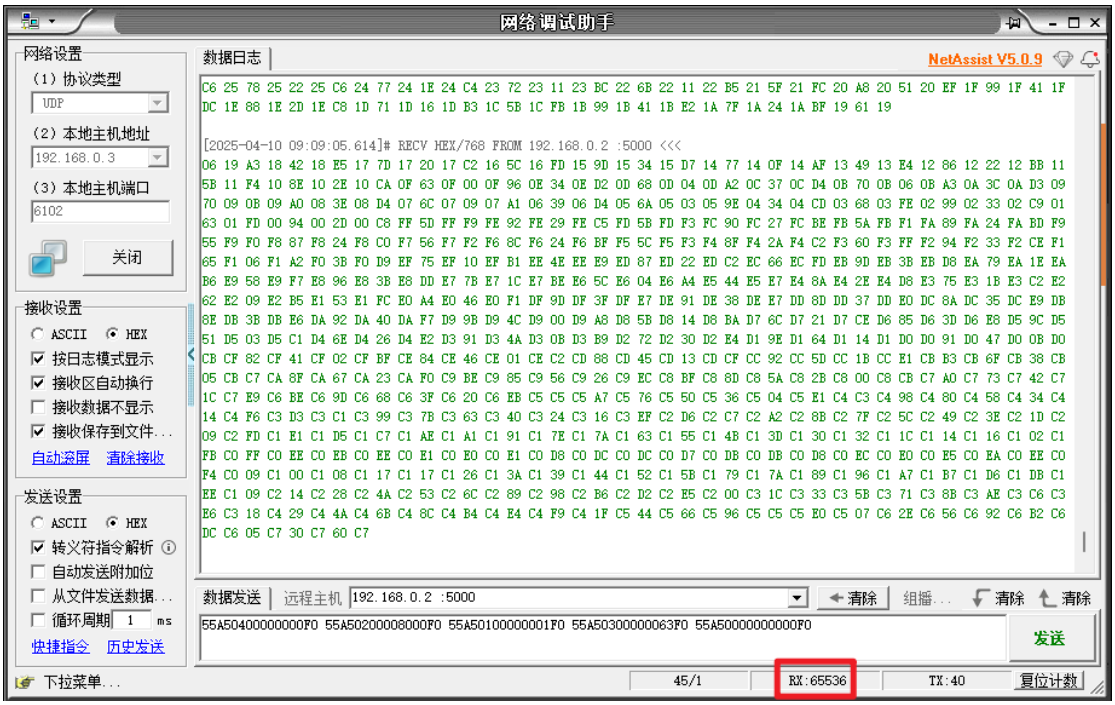


图 1-30 网络调试助手接收数据

1.4.6 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。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 1-31 所示，得到的波形图如下图 1-32 所示。

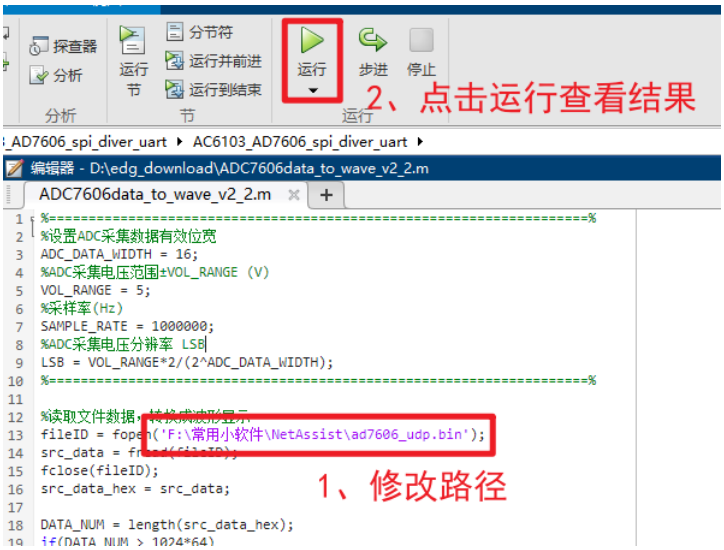


图 1-31 修改文件路径并运行

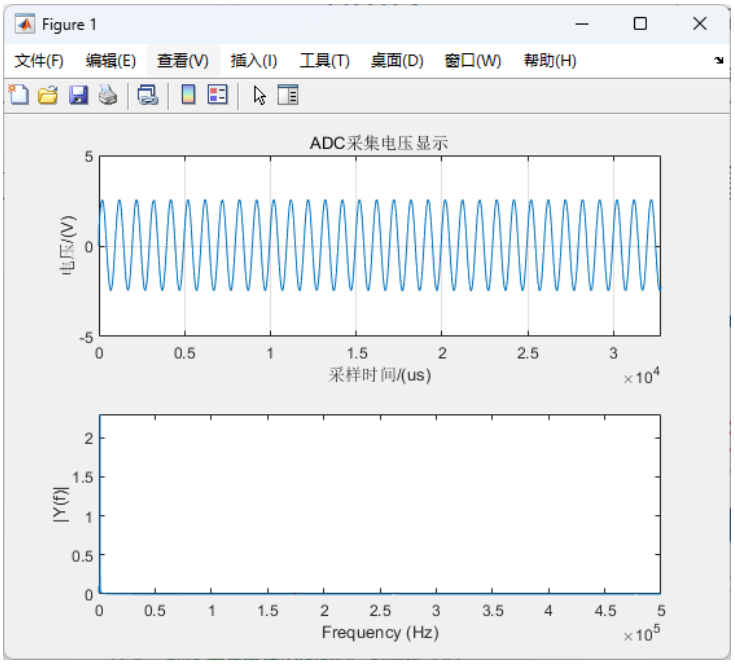


图 1-32 MATLAB 分析波形图

前面我们提到过本次实验提供的信号源为 1Khz，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。

1.4.7 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“小梅哥数据采集仪”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，软件获取位置如下：

https://www.corecourse.cn/forum.php?mod=viewthread&tid=29814&highlight=%E4%B8%8A%E4%BD%8D%E6%9C%BA

双击 QSCOPE_1.0.14.exe 打开上位机软件，软件初始化界面如下所示。

图 1-33 上位机软件初始化

本次实验使用该软件的方式如下所示：

1、设置通信方式，选择以太网通信，操作如下图 1-34 所示。

图 1-34 设置通信方式

2、选择 ADC 型号，操作如下图 1-35 所示。

图 1-35 选择 7606C

3、接着点击“连接按钮”，操作如下图 1-36 所示。

图 1-36 点击连接按钮

4、接着点击软件右下角的“点击开始采样”按钮，如下图 1-37 所示。

图 1-37 点击开始采样

此时可以看到波形界面已经绘画除了波形，因为设置的采样数量默认为 4K（4096），所以获取到的数据为 8192 个字节的数据（以太网一次接收 8 位的数

据），如下所示

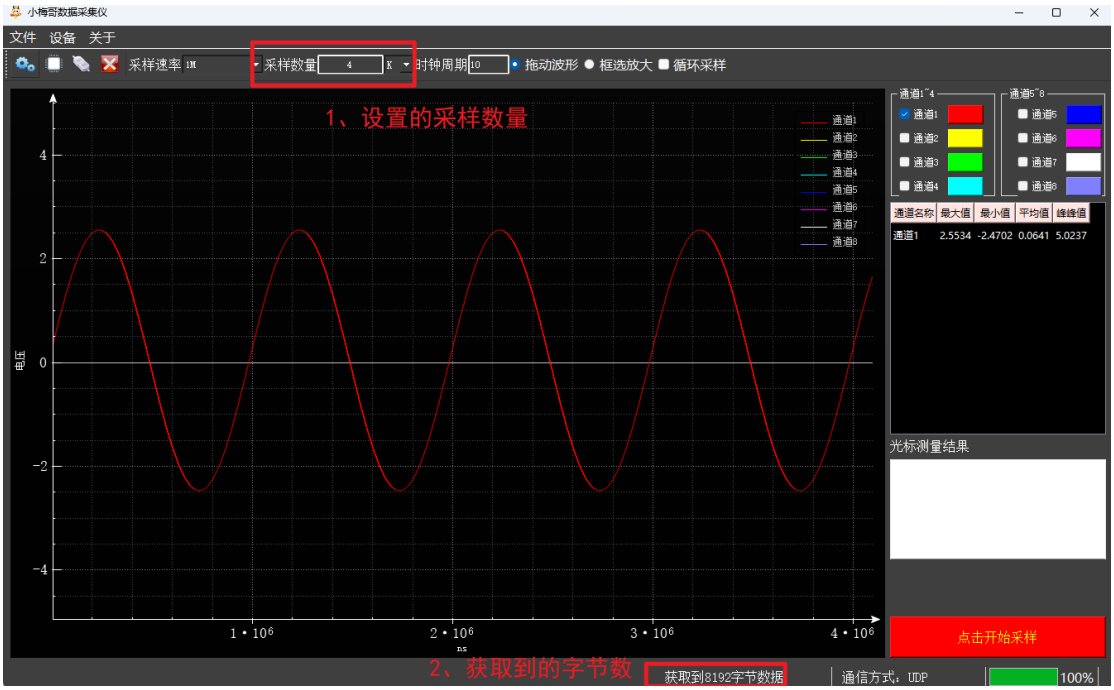


图 1-38 采集到的数据

接着将采样数量改成 8K，重新进行采样，采样结果如下所示。

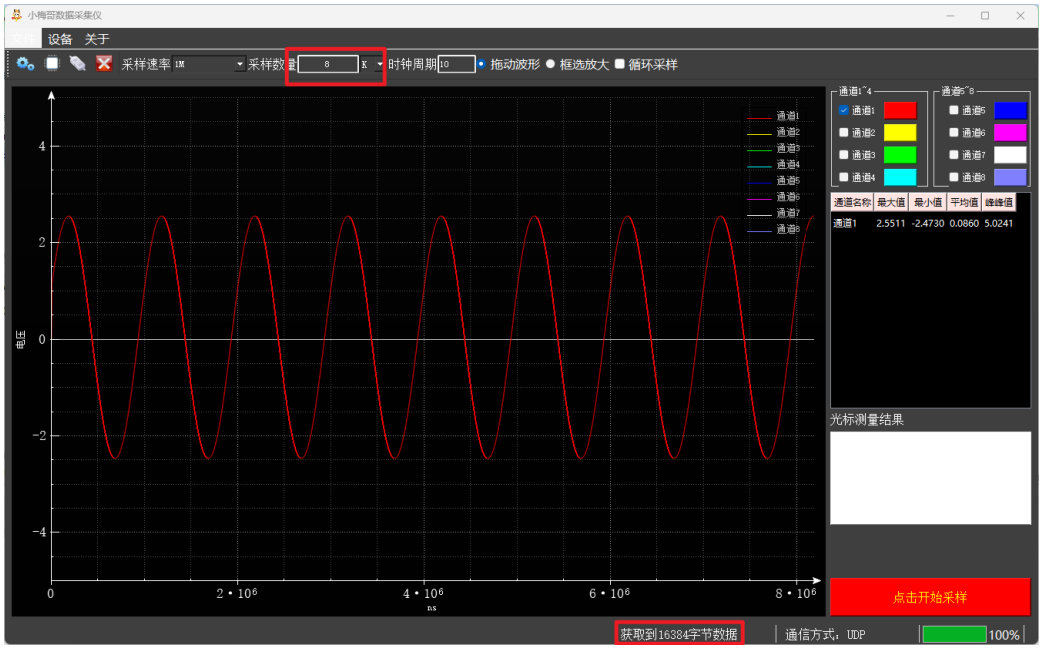


图 1-39 采集 8K 数据

接着通过软件计算出波形的频率，在波形界面右击，选择“设置光标”，如下图所示。

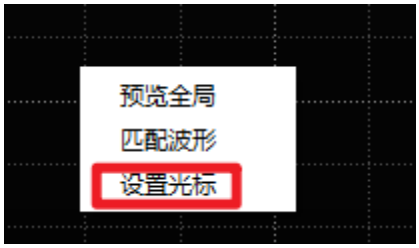


图 1-40 设置光标

在弹出的对话框中使能光标 1，参考通道选择通道 1，最后点击“确定”按钮。

图 1-41 设置光标 1

接着在波形合适的位置单机即可固定住光标，如下图所示。

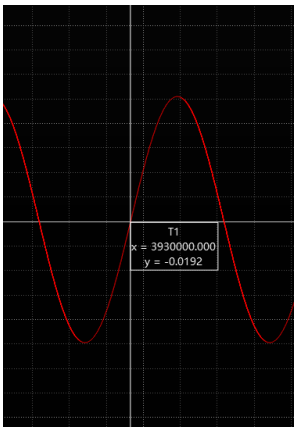


图 1-42 固定光标 1

接着再添加一根光标，使能光标 2，参考通道选择通道 1，点击“确定”按钮，最后将光标 2 固定在合适的位置，可以在光标测量结果框中显示出波形频率，如下所示。

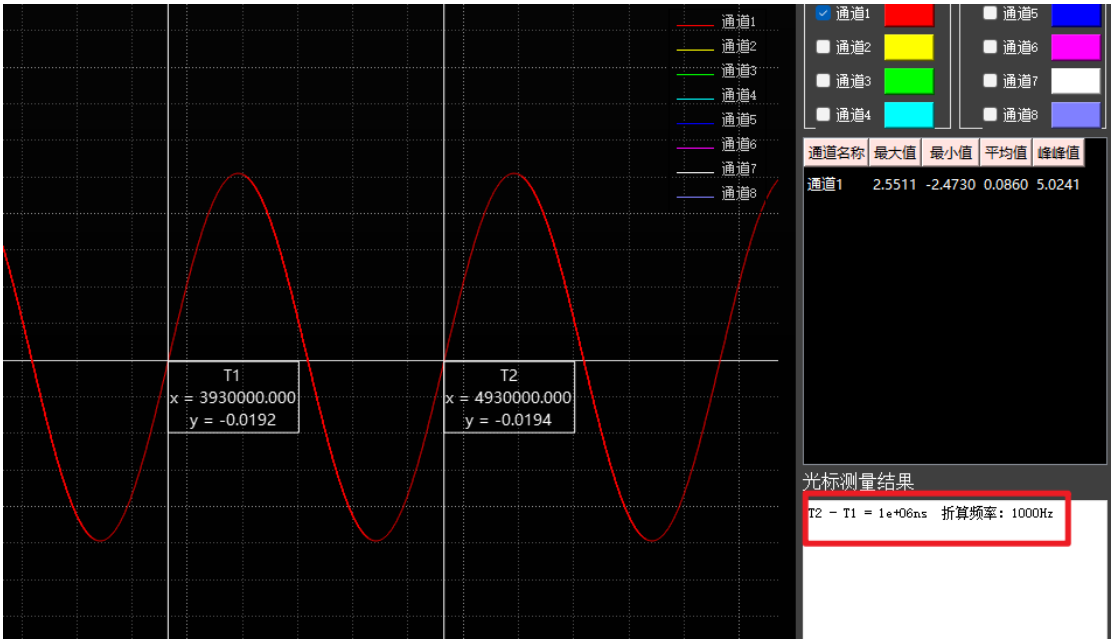


图 1-43 频率测量

至此，基于千兆以太网的 AD7606C 数据采集 DDR3 存储系统已经验证完成。

1.5 思考与总结

本次实验介绍了基于 AD7606C 的千兆以太网收发，用户通过网口调试助手向开发板发送指令数据配置 AD7606C 的五个寄存器，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。

本次实验涉及时钟域的转换，以及采集数据位宽的转换，完成这些转换需要对 FIFO 有一定的了解，本次实验也要求读者对以太网协议有足够的认识，想要进一步理解以太网功能原理可参看前面以太网相关章节的内容。