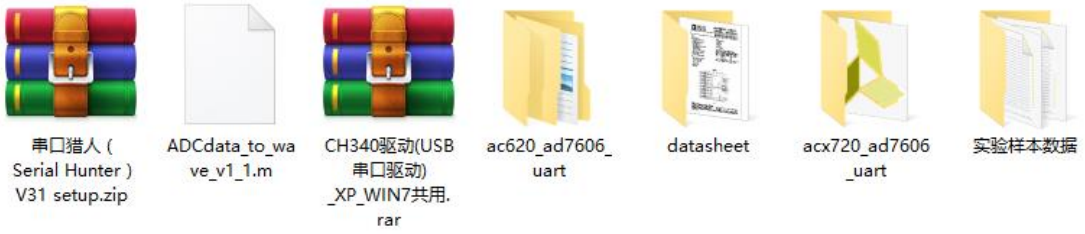
基于 AD7606 的多通道数据采集系统

文件说明：

本次实验配套的文件压缩包名为 ad7606_uart.rar，解压后可得到配套的工程文件，采样数据和调试工具等，各文件功能如下所示：



1. acx720_ad7606_uart：基于 ACX720 FPGA 开发板（Artix-7 FPGA）的工程源码
2. ac620_ad7606_uart：基于 AC620 FPGA 开发板（Cyclone IV E FPGA）的工程源码
3. 实验样本数据：笔者实验时得到的几组采样结果样本数据
4. ADCdata_to_wave_v1_1.m：基于 MATLAB 的采样结果数据处理函数
5. CH340 驱动(USB 串口驱动)_XP_WIN7 共用.rar：CH340 串口驱动程序
6. 串口猎人 (Serial Hunter) V31 setup.zip：串口调试软件

概述

本案例基于 Xilinx Artix-7 系列 FPGA，结合 ADI 公司的 16 位 8 通道并行采样 ADC 芯片，实现了对 AD7606 型 8 通道 16 位 ADC 的数据转换控制并输出，数据采集完成后存储在 FIFO 中，并通过串口传输，用户可以在电脑上通过串口调试工具对采样到的数据进行查看并通过指令对 ADC 芯片进行配置，控制其采样频率，数据采样个数以及通道数，并将输出的数据导出，进行进一步的数据处理分析。

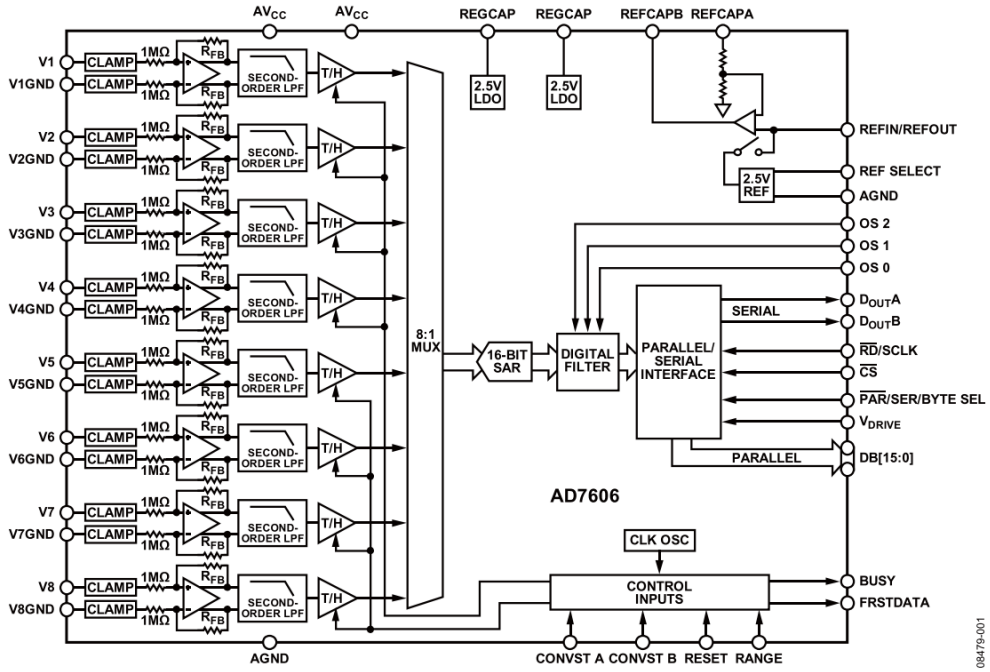
系统原理

ad7606 概述

本案例使用的是 ADI 公司的 16 位 8 通道同步采样模数转换器 AD7606，AD7606 是 16 位 8 通道同步采样模数数据采集系统（DAS）。内置模拟输入箝位保护、二阶抗混叠滤波器、跟踪保持放大器、16 位电荷再分配逐次逼近型模数转换器（ADC）、灵活的数字滤波器、2.5V 基准电压源、基准电压缓冲以及高速串行和并行接口。AD7606 采用 5V 单电源供电，可以处理±10V 和±5V 真双极性输入信号，同时所有通道均能以高达 200kSPS 的吞吐速率采样。输入箝位保护电路可以耐受最高达±16.5V 的电压。无论以何种采样频率工作，其模拟输入阻抗均为 1MΩ。采用单电源工作方式，具有片内滤波和高输入阻抗，因此无需驱动运算放大器和外部双极性电源。AD7606 抗混叠滤波器的 3dB 截止频率为 22kHz；当采样频率为 200kSPS 时，它具有 40dB 抗混叠抑制特性。

芯片对外提供 spi 和并行的数字接口。当 AD7606 的 8 个通道全部以 200KSPS 的最高速率进行转换时，数据输出速率达到了 25.6Mbps，需要使用高性能 MCU 的 SPI 外设才能勉强满足该速率要求。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 ad7606 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 AD7606 时，可以通过在 FPGA 上设计 ad7606 的转换控制逻辑，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，也可以直接使用软和控制器完成数据的处理和传输工作，如典型的，在 Intel FPGA 器件上使用 NIOS II 软核控制器，将 AD7606 的转换结果通过串口或以太网传输到电脑上，也可以直接将数据显示在液晶显示屏上。

功能框图

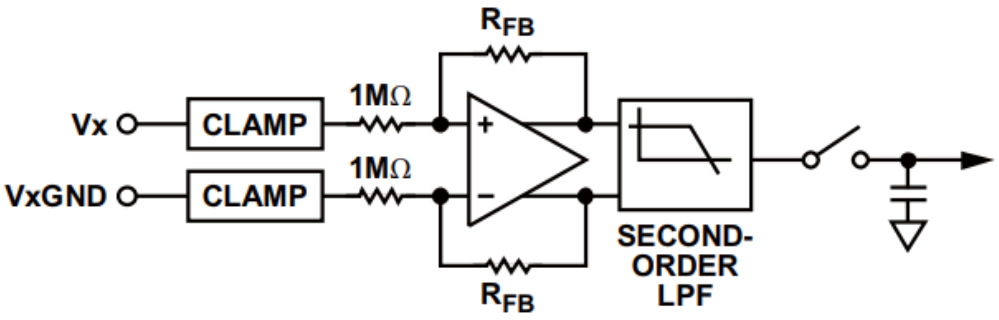


AD7606 芯片的功能框图如上图所示，采集到的数据在经过稳压滤波和采样保持后通过 8 选 1 多路选择器被分别送入到 16 位逐次逼近型 ADC 芯片 AD7606 中进行转换，最后经由数字滤波后输出，当数据是以串行模式输出时，数据会从 DoutA、DoutB 中输出，如果数据是以并行方式输出，那么数据将从 DB[15:0]中输出，下面为 AD7606 部分模块的说明介绍。

模拟输入

AD7606 可处理真双极性、单端输入电压。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。AD7606 的模拟输入阻抗为 1MΩ。这是固定输入阻抗，不随 AD7606 采样频率而变化。AD7606 的模拟输入结构如

下图，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±16.5V。

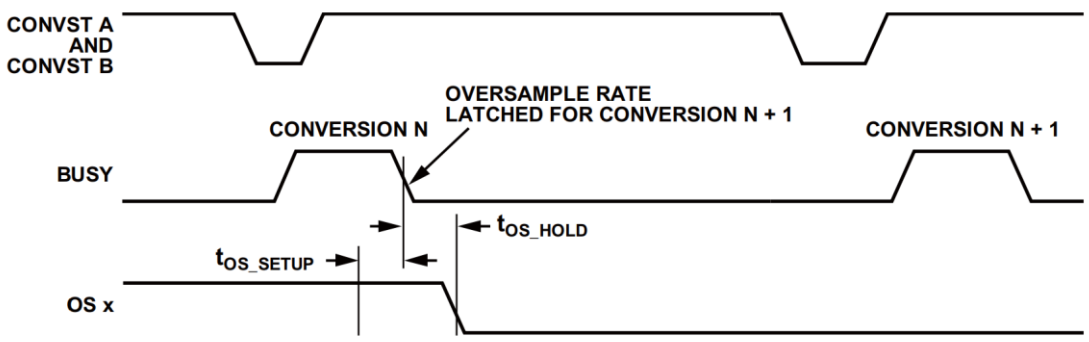


数字滤波器与过采样

AD7606 内置一个可选的数字一阶 sinc 滤波器，在使用较低吞吐率或需要更高信噪比或更宽动态范围的应用中，应使用该滤波器。数字滤波器的过采样倍率有过采样引脚 OS[2:0] 控制（具体参考 AD7606 数据手册）。OS 2 为 MSB 控制位，OS 0 为 LSB 控制位，下表提供了用来选择不同过采样倍率的过采样位解码。

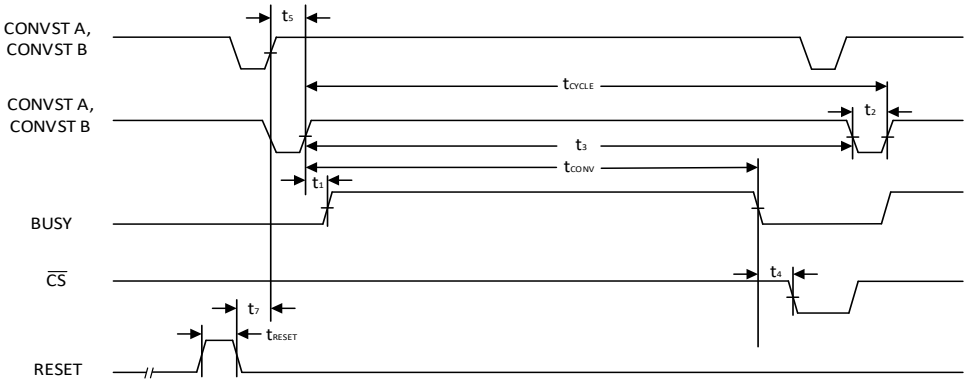
OS[2:0]	过采样倍率	5 V范围SNR(dB)	10 V范围SNR(dB)	5 V范围3 dB带宽(kHz)	10 V范围3 dB带宽(kHz)	最大吞吐量CONVST频率(kHz)
000	No OS	89	90	15	22	200
001	2	91.2	92	15	22	100
010	4	92.6	93.6	13.7	18.5	50
011	8	94.2	95	10.3	11.9	25
100	16	95.5	96	6	6	12.5
101	32	96.4	96.7	3	3	6.25
110	64	96.9	97	1.5	1.5	3.125
111	无效					

OS 引脚在 BUSY 下降沿锁存，从而设置下一个转换的过采样倍率（如下图所示），如果 OS 引脚选择过采样倍率 8，则下一个 CONVST x 上升沿采集各通道的第一个样点，一个内部产生的采样信号采集所有通道的其余 7 个样点，然后对这些样点求平均值，以改进 SNR 性能。开启过采样时，CONVST A 和 CONVST B 引脚必须连在一起驱动，转换过程中 BUSY 保持高电平的时间会延长。BUSY 保持高电平的实际时间取决于所选的过采样倍率，过采样倍率越高，则 BUSY 保持高电平的时间或总转换时间越长。

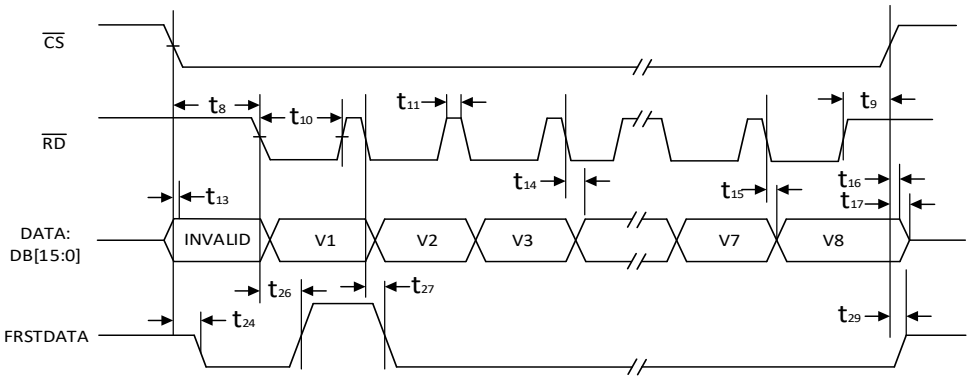


工作时序

AD7606 根据采样方式不同具有多种驱动时序，本次采用的为并行输出（即 8 个 16 位的数据通过 16 根并行线一个接着一个输出），转化后读取模式。其时序图有两部分组成：1.完成 AD 转化；2.读取 AD 数据。其中的时间可以参考 ADI 公司的手册。当 CONVST A 和 CONVST B 通道都变为上升沿时，BUSY 信号转变为高电平，代表转换开始，直到 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。并行工作模式下，当 $\overline{CS}$ 和 $\overline{RD}$ 都为低电平时，会使能总线，将转换结果输出到并行数据总线上，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。并行模式下，每次数据的输出为 16 位，对应一个通道。



CONVST 时序——转换之后读取



并行模式，独立的 $\overline{CS}$ 和 $\overline{RD}$ 脉冲

控制器说明

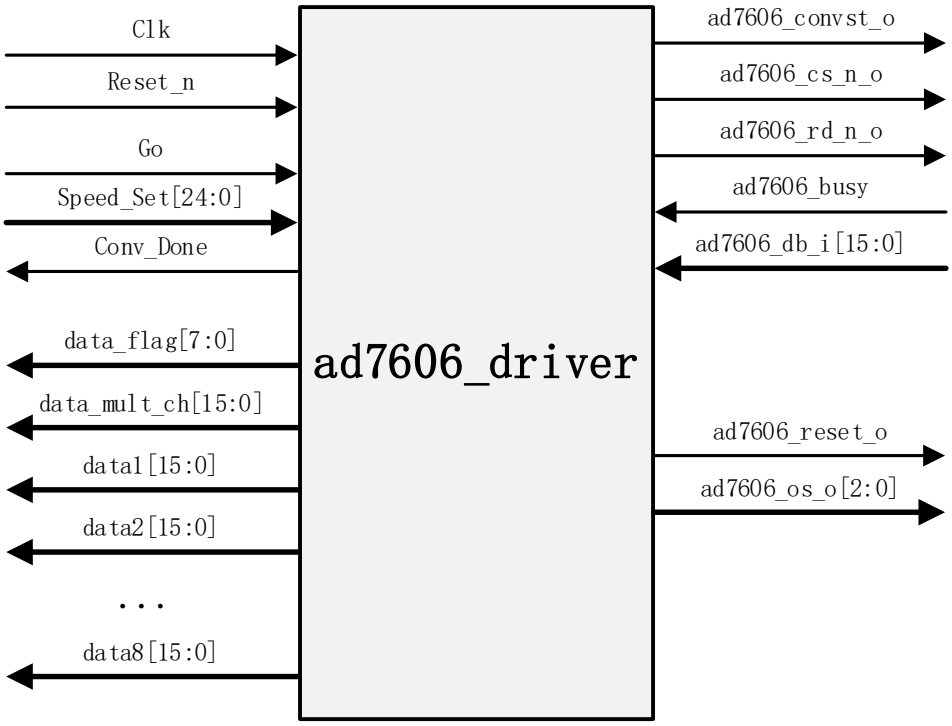
功能简介

该控制器实现了对 AD7606 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 AD7606 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。我们还为该模块提供了基于 quartus II 的数据采集方案,有关基于 quartus II 的数据采集方案与本次设计的基于 vivado 的数据采集系统原理一致，只是应用的平台不同，用户可以参考我们提供的相关源工程。

接口说明

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 AD7606 控制器与 AD7606 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，即可快速高效的使用该控制器来控制 AD7606 芯片完成数据转换。



类	信号名	位	功能简介
	Clk	1	系统时钟，为了让采样速率准确，要求为 50MHz

	Reset_n	1	系统复位，低电平复位
控制信号	Go	1	采样使能信号，为高电平就使能采样，低电平则在已经开始的一轮采样结束后，停止下一次采样。
	Speed_Set	25	采样速率控制端口，Speed_Set = 1000000000/20/speed - 1
	Conv_Done	1	一次采样完成标志信号，单时钟周期脉冲信号。每次 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	8	转换结果有效标志信号，因为 AD7606 有 8 个通道，转换结果是依次输出，并非同时的，所以设置 8 个 Flag 信号，每个通道的结果就绪之后，就产生一个 Flag 信号，通知外部可以取用。另外，如果只关心其中的部分通道，则只需要关心 data_flag 中对应的位即可。
	data_mult_ch	16	多通道数据输出端口，该通道 16 位，在不同的时刻，输出不同通道的转换结果，使用时，与 data_flag 信号配合，data_flag 的哪一位出现高脉冲，则代表当前 data_mult_ch 的值为该通道的转换结果。该端口设计的目的是用于往 FIFO、RAM 等存储器中存储结果时使用。
	data1..8	16	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。这些端口主要用于每个通道的数据需要分别应用的场合。
ADC 芯片控制信号	ad7606_cs_n_o	1	ad7606 芯片选中控制信号，可以从 AD7606 中读取转换结果时，需要使该信号为低电平。
	ad7606_rd_n_o	1	ad7606 转换结果读取信号，该信号的下降沿， AD7606 将特定通道的采样结果送到 16 位数据线上，供外部读取。外部可以在 rd_n 信号的上升沿读取 16 位数据线上的结果。
	ad7606_busy_i	1	ad7606 转换状态标志信号，为高电平则表明 ad7606 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 ad7606 中的数据。
	ad7606_db_i	16	ad7606 的 16 位数据线，读取时，输出对应通道的转换结果。

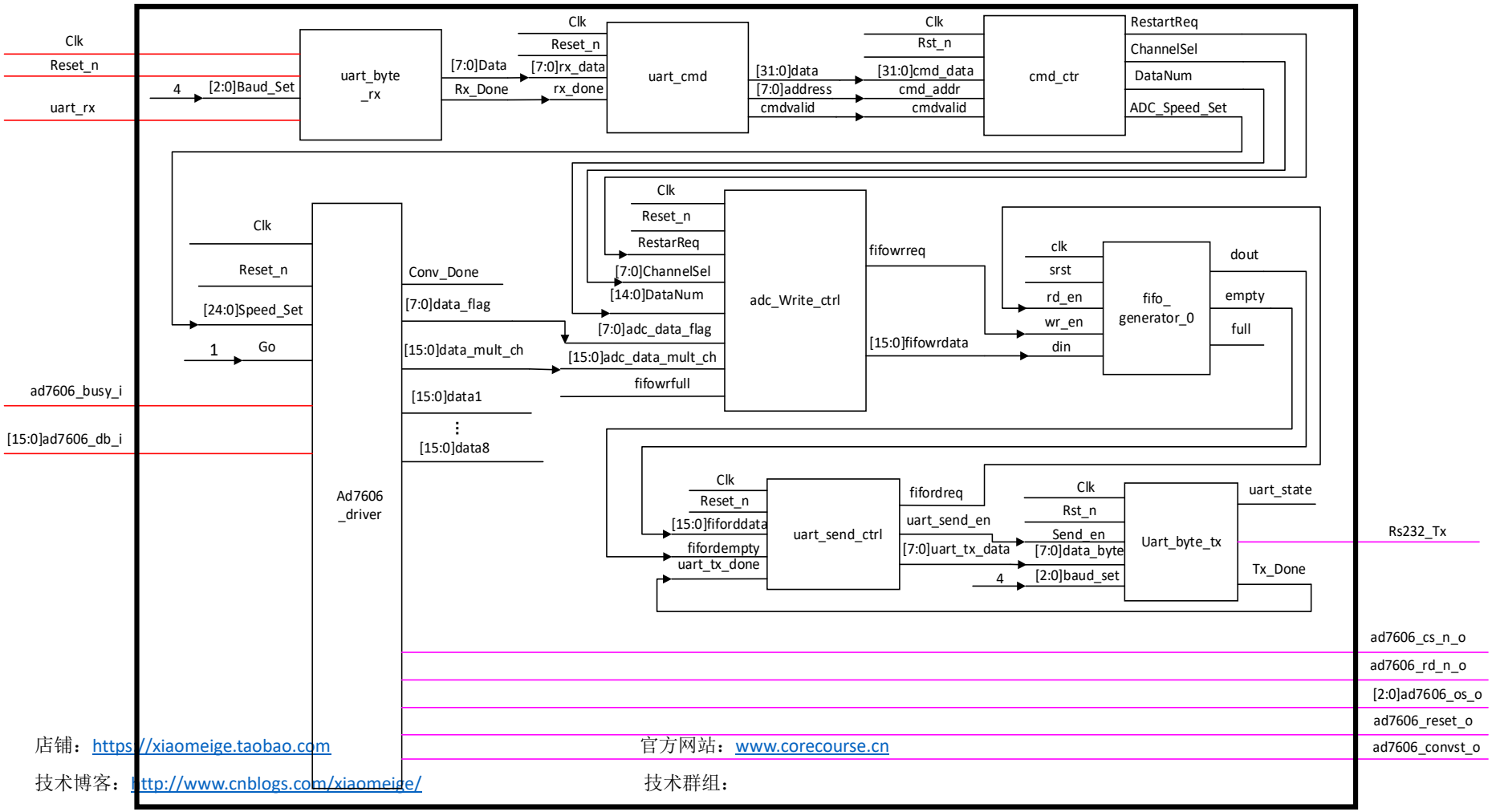
	ad7606_os_o	3	ad7606 过采样控制信号，使用过采样可以进一步提高 ad7606 的采样精度，使用过采样会降低 ad7606 的有效转换速率，关于过采样的功能和使用方法，可以参考 ad7606 的 datasheet，默认为 0，则表示不使用过采样。能够运行在最高的转换速率（200Ksps）
	ad7606_reset_o	1	ad7606 的复位信号，复位 ad7606 内部各个功能单元的工作状态。
	ad7606_convst_o	1	ad7606 转换开始信号，该信号的上升沿启动 ad7606 内部的采样转换逻辑开始新一轮的采样转换。

应用案例

实验原理图

本次实验原理图如下，其中 Reset_n 信号和 FIFO 的 srst 信号都为复位信号 Rst_n，波特率设置为 4，根据查找表可知为 115200bps，GO 信号为 ADC 控制信号，用户可通过该信号直接控制 ADC 运行状态，这里为了实验方便直接将其置 1，红色为输入信号，粉色为输出信号，其余信号的连接与关系如图所示：

AD7606_UART


Clk
Reset_n
uart_rx
4
[2:0]Baud_Set
uart_byte
_rx
[7:0]Data
Rx_Done
Clk
Reset_n
[7:0]rx_data
rx_done
uart_cmd
[31:0]data
[7:0]address
cmdvalid
Clk
Rst_n
[31:0]cmd_data
cmd_addr
cmdvalid
cmd_ctr
RestartReq
ChannelSel
DataNum
ADC_Speed_Set
Clk
Reset_n
RestarReq
[7:0]ChannelSel
[14:0]DataNum
adc_Write_ctrl
fifowrreq
[15:0]fifowrdata
clk
srst
rd_en
wr_en
din
fifo_
generator_0
dout
empty
full
Clk
Reset_n
[24:0]Speed_Set
1
Go
Conv_Done
[7:0]data_flag
[15:0]data_mult_ch
[15:0]data1
[15:0]data8
[7:0]adc_data_flag
[15:0]adc_data_mult_ch
fifowrfull
ad7606_busy_i
[15:0]ad7606_db_i
Ad7606
_driver
Clk
Reset_n
[15:0]fiforddata
fifordempty
uart_tx_done
uart_send_ctrl
fifordreq
uart_send_en
[7:0]uart_tx_data
4
Clk
Rst_n
Send_en
[7:0]data_byte
[2:0]baud_set
Uart_byte_tx
uart_state
Rs232_Tx
Tx_Done
ad7606_cs_n_o
ad7606_rd_n_o
[2:0]ad7606_os_o
ad7606_reset_o
ad7606_convst_o

模块介绍

AD7606 控制器驱动模块（Ad7606_driver）

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，控制其将转换完成数据 data_mult_ch 写入 FIFO 或者 RAM 的同时，也指出了该信号来自哪个通道。通过这两个信号，我们可以实现让 ADC 以特定的采样速率多次采样一个或多个通道的数据。每当 data_flag 中任意一位为 1，则将 data_mult_ch 中的值写入 FIFO 或者 RAM 中。由于 FIFO 和 RAM 等存储器，只有一个数据输入接口，所以这个时候用一个 data_mult_ch 端口分时输出不同通道的采样结果，就比使用 data1~data8 这 8 个 16 位的数据端口分别输出各自通道的采样结果要方便。

例如， 将通道 1、2、5、8 的采样结果存入 FIFO。就可以使用下面的写法。

```
module adc_wr_fifo(
    input Clk,
    input Reset_n,
    input [7:0]data_flag;
    input [15:0]data_mult_ch;
    output reg fifo_wrreq,
    output reg [15:0]fifo_data
);

    always@(posedge Clk or negedge Reset_n)
    if(!Reset_n)begin
        fifo_wrreq <= 0;
        fifo_data <= 0;
    else if(data_flag == 8'b1001_0011)begin
        fifo_wrreq <= 1'd1;
        fifo_data <= data_mult_ch;
    end
    else begin
        fifo_wrreq <= 0;
        fifo_data <= fifo_data;
    end

endmodule
```

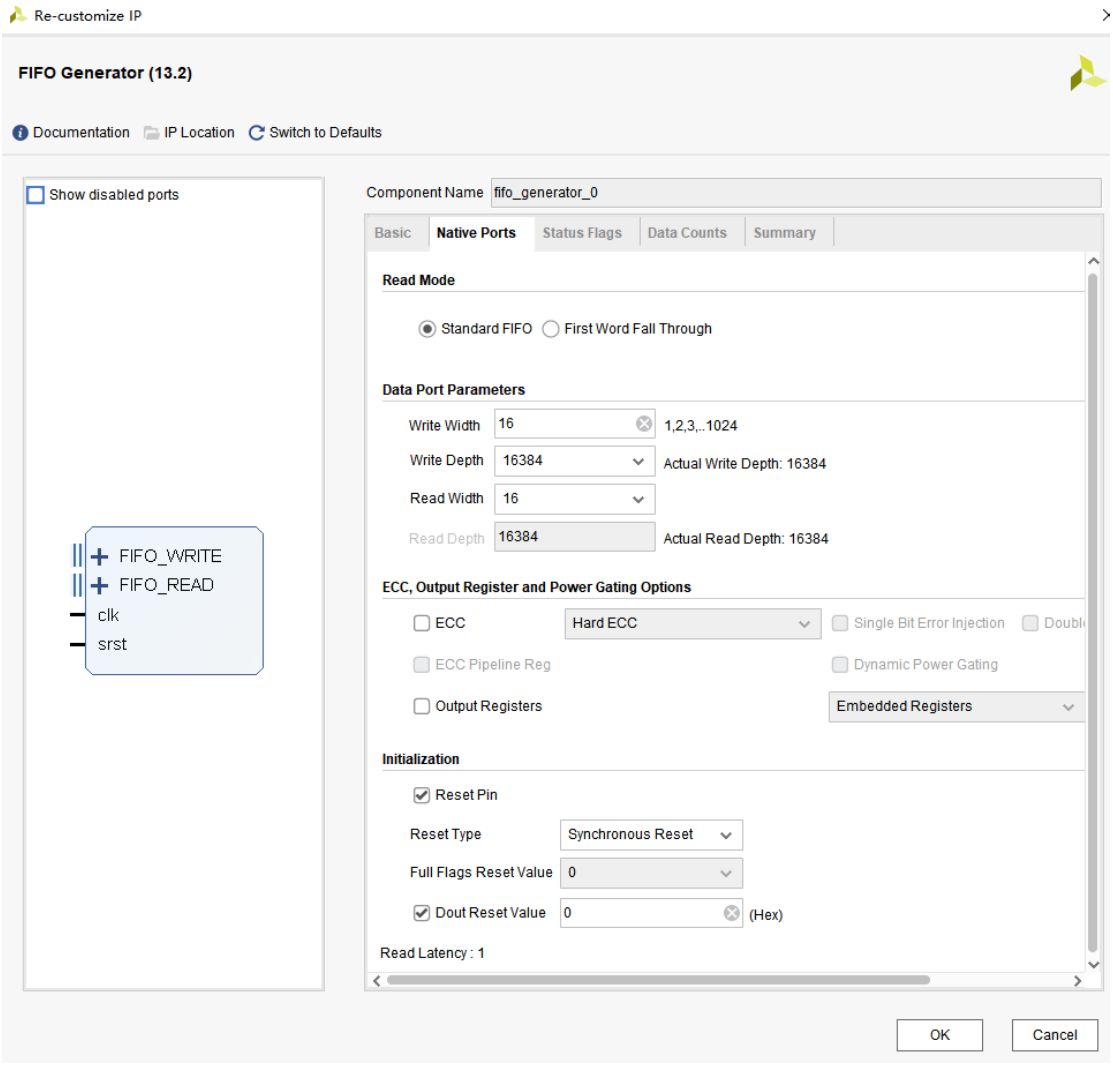
ADC 采集控制模块（adc_Write_ctrl）

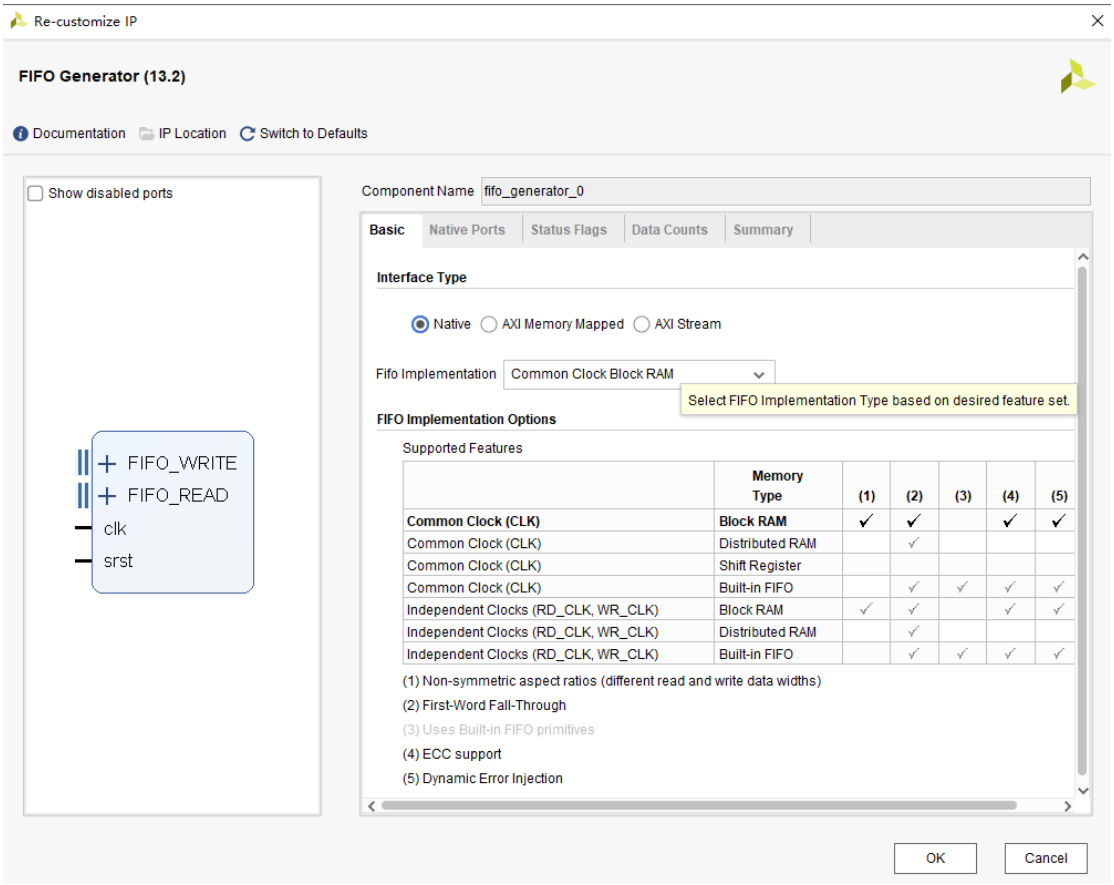
根据指令控制 ADC 进行相应的数据采样，如采集深度，通道选择，在采样完成后生成一个 FIFO 写请求信号并将采样到的数据传送至 FIFO 中。

```
//采样控制逻辑，每次采样请求信号到来开始采样，采样个数满了停止采样。
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)
    sample_en <= #1  1'b0;

//采样个数计数器，在采样使能阶段，每个 flag 到来的时候计数器自加1
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;

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

always@(posedge Clk)
    fifowrdata <= #1  adc_data_mult_ch;
```

FIFO IP 核（fifo_generator_0）

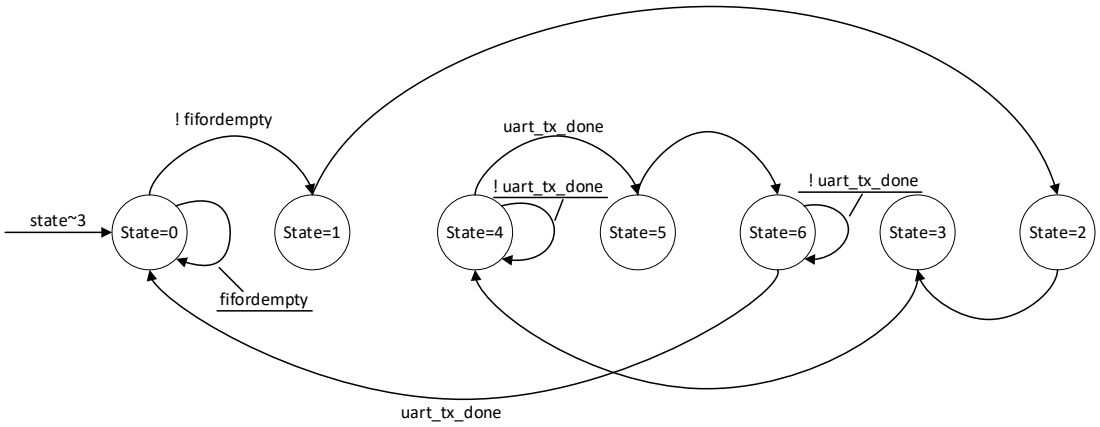
当 IP 核接收到来自 adc_Write_ctrl 的写请求信号时，ADC 采集到的数据被存入 FIFO 中缓冲，当接收到来自 uart_send_ctrl 的读请求信号时，数据就被读出，并经由串口发送模块发送到主机上。这里 FIFO 读写位宽都设置的为 16 位，数据深度为 16384，共用同一个时钟，读模式选择为 standard FIFO，接口类型设置为 native。


Re-customize IP
FIFO Generator (13.2)
Documentation
IP Location
Switch to Defaults
Show disabled ports
FIFO_WRITE
FIFO_READ
clk
srst
Component Name
fifo_generator_0
Basic
Native Ports
Status Flags
Data Counts
Summary
Read Mode
Standard FIFO
First Word Fall Through
Data Port Parameters
Write Width
16
1,2,3,..1024
Write Depth
16384
Actual Write Depth: 16384
Read Width
16
Read Depth
16384
Actual Read Depth: 16384
ECC, Output Register and Power Gating Options
ECC
Hard ECC
Single Bit Error Injection
ECC Pipeline Reg
Dynamic Power Gating
Output Registers
Embedded Registers
Initialization
Reset Pin
Reset Type
Synchronous Reset
Full Flags Reset Value
0
Dout Reset Value
0
(Hex)
Read Latency : 1
OK
Cancel



串口发送控制模块（uart_send_ctrl）

该模块主要负责控制串口的发送，通过向 FIFO 发送读请求信号将 FIFO 中缓冲的数据读取出来，通过向串口输出发送使能信号 uart_send_en 来控制串口数据的发送，由于串口一次只能发出 8 位数据，而从 FIFO 中读取的数据为 16 位，所以每读取一次数据，串口需要传输两次，这里我们通过一个状态机来实现对串口数据发送的控制，其状态图如下图所示



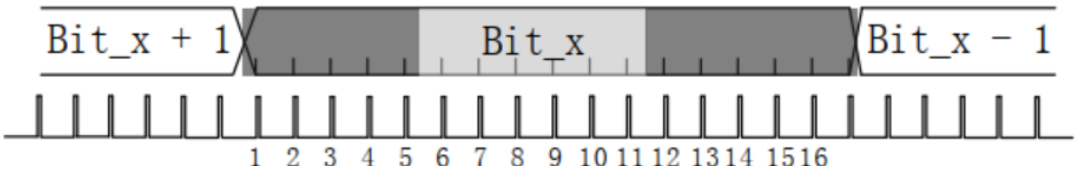
串口发送模块（Uart_byte_tx）

该模块负责将读取的数据发送到主机上，当接收到来自发送控制模块的 uart_send_en 发送使能信号，串口开始接受从串口发送控制器中传送过来的数据，并在发送完成后生成一个时钟周期高电平的发送完成信号 Tx_Done，串口发送控制器在接收到该信号后便会控制串口进行下一次传输，同时为了保证模块的复用性，模块设置了 5 种不同的波特率，用户可以根据自己的需求选择不同波特率，这里为了实验的方便，我们在顶层将其设置为了 115200bps。

```
always@(posedge Clk or negedge Rst_n)
if(!Rst_n)
    bps_DR <= 16'd5207;
else begin
    case(baud_set)
        0:bps_DR <= 16'd5207;
        1:bps_DR <= 16'd2603;
        2:bps_DR <= 16'd1301;
        3:bps_DR <= 16'd867;
        4:bps_DR <= 16'd433;
        default:bps_DR <= 16'd5207;
    endcase
end
```

串口接收模块（uart_byte_rx）

该模块主要负责接收来自主机的数据，并在每次数据接收完成后生成一个接收完成信号 Rx_Done 输出到接收转命令模块，同样的这里也有个波特率设置模块，为了确保采样结果的准确性每个数据都会进行 6 次采样，如果采样累加结果大于 4 则说明该值为 1，反之则为 0，该部分代码不理解的可以参考《FPGA 系统设计与验证实战指南》中的串口接收模块设计与验证一节。



接收转命令模块（uart_cmd）

在讲解该模块之前，需要先对配置 AD7606 的寄存器进行说明，配置 AD7606 的寄存器一共有四个，他们的功能和地址分别如下：

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位为 1，则该通道的采样结果就会被存入 FIFO 并通过串口发送。
DataNum	2	16	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，而且设置了 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 200Ksps，所以可以通过设置该寄存器的值来让 ADC 的采样速率在 1~200Ksps 范围内调整，以适应不同的应用场景。该寄存器的值与采样速率关系为：ADC_Speed_Set = 1000000000/20/speed – 1，其中 speed 就是实际要设置的采样速率。

串口一次发送的数据内容为 1 个字节，为了实现通过串口修改这些寄存器的值，需要发送多个字节才能实现，为此，设计了简单的串口数据帧，该帧一帧数据共 8 个字节，包含帧头、帧尾、地址段、数据段。帧格式如下所示：

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

每帧数据共 8 个字节，分别用 D0~D7 表示，其中，D0 和 D1 两个数据作为帧头，其值分别为固定的 0x55、0xA5，D7 作为帧尾，其值为固定的 0xF0。D2 则为要操作的寄存器地址，D3 为要写入寄存器的数据的 24~31 位，D2 为要写入寄存器的数据的 16~24 位，D2 为要写入寄存器的数据的 8~15 位，D0 为要写入寄存器的数据的 0~7 位。

而该模块的作用就是将串口接收到的数据打包成该格式，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 AD7606 进行相对应的配置。该模块在执行过程中会对数据进行判定，当数据符合 D0 为 8'd55,D1 为 8'dA5,D7 为 8'dF0，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。

```
always@(posedge Clk or negedge Reset_n)
if(!Reset_n) begin
    address <= #1 0;
    data <= #1 0;
    cmdvalid <= #1 0;
end else if(r_rx_done)begin
    if((data_str[0] == 8'h55) && (data_str[1] == 8'hA5) &&
(data_str[7] == 8'hF0))begin
        data[7:0] <= #1 data_str[6];
        data[15:8] <= #1 data_str[5];
        data[23:16] <= #1 data_str[4];
        data[31:24] <= #1 data_str[3];
        address <= #1 data_str[2];
        cmdvalid <= #1 1;
    end
end
else
    cmdvalid <= #1 0;
```

指令转控制模块（cmd_ctr）

该模块的作用旨在将从串口转指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，每当接收到 cmdvalid 信号时，该模块便通过地址对判断对那个寄存器执行操作，并从数据中提取对应的数据将其输出到相应的寄存器

```
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;
end
else if(cmdvalid)begin
    case(cmd_addr)
        0: RestartReq <= 1'b1;
        1: ChannelSel <= cmd_data[7:0];
        2: DataNum <= cmd_data[15:0];
        3: ADC_Speed_Set <= cmd_data;
        4:
            begin
                ChannelSel <= cmd_data[7:0];
                DataNum <= cmd_data[23:8];
                RestartReq <= 1'b1;
            end
      default:;
  endcase
end
else
    RestartReq <= 1'b0;
```

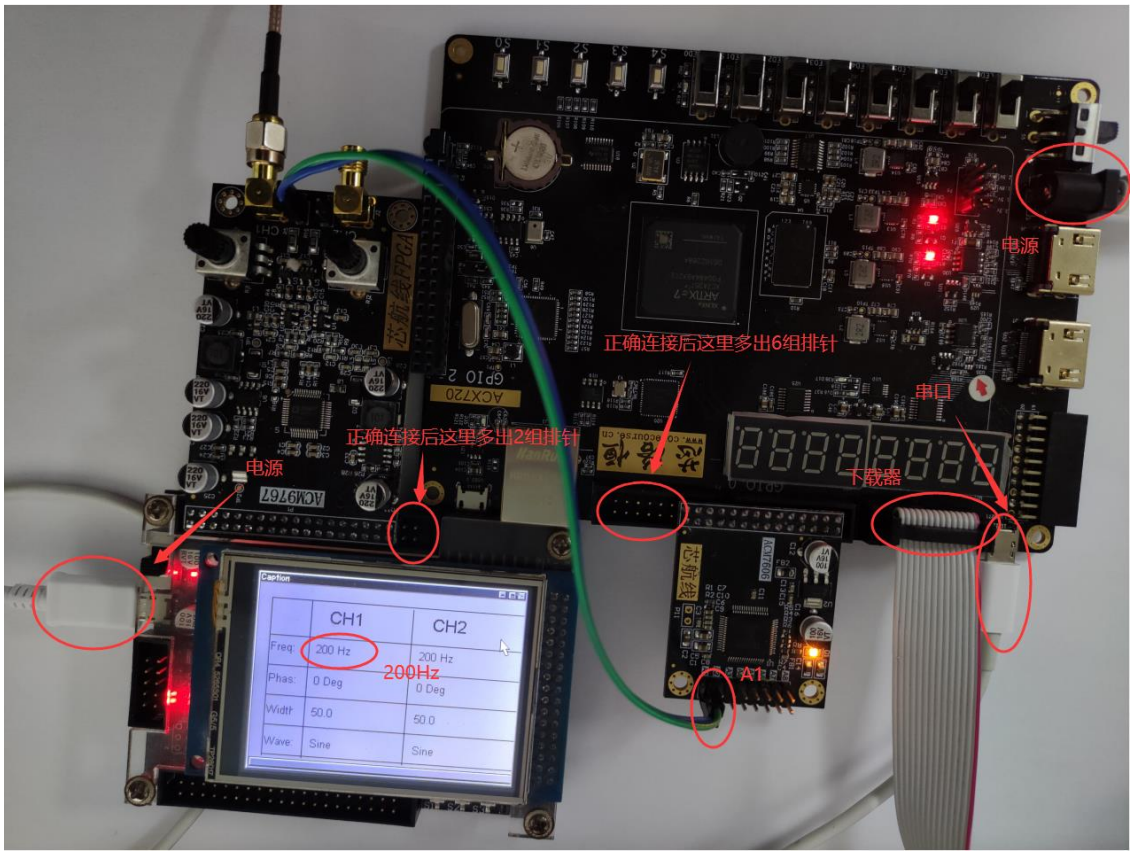
本次实验是以串口接收发送为基础，通过添加 AD7606 等模块以实现想要的功能，用户可以运行本次试验提供的例程，查看相关的程序源码对本次实验各个模块的关联与作用进行更深入的了解，接下来便开始对本次实验进行验证。

工程演示

硬件连接

要验证本次实验，首先第一步就是对本次实验的整个系统硬件进行连接，本次工程硬件具体的连接如下图所示，这里我们用 9767 自制了一个信号源，分别输出 200Hz 的正弦波、

方波、三角波，用户可以直接使用信号发生器作为信号源，注意串口必须连接在 ACX720 板子的左侧接口，杜邦线连接的是 A1 端口，这里强调一点，AD7606 模块在正确连接后右边会多出 6 组排针，由于视觉缘故，这里很容易连接错误，连接错误后串口猎人上容易出现有发送而无接收的情况，电源线与下载器连接如图中所示，在硬件连接好后打开开关就可以下载程序了，用户可以直接将我们提供的 bit 文件夹下的文件烧写进板中，也可打开工程一步步下载。



配置 ADC

下载完成后开发板上会有两个红灯亮起，接着便可以对 AD7606 进行配置了，首先打开串口猎人，注意看有没有检测到串口，如果没有就看看是不是串口线连接有问题或者是否没装驱动，接着在波特率一栏设置其为 115200，接着打开高级发码界面，以上面接收转命令模块中介绍到的数据帧格式对 AD7606 的四个寄存器进行配置。

例如，PC 端要设置采样数据个数(DataNum 寄存器，地址为 2)为 16000（0x3E80）个，则发送的数据帧内容为：0x55 0xA5 0x02 0x00 0x00 0x3E 0x80 0xF0。

例如，PC 端要设置采样速率(ADC_Speed_Set 寄存器，地址为 3)为 100K，则对应的 ADC_Speed_Set 值为 1000000000/20/100000 – 1 = 499（0x000001F3）个，则发送的数据帧内

容为：0x55 0xA5 0x03 0x00 0x00 0x01 0xF3 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，当然如果在进行第二次采样时配置与第一次相同，也可以只配置 0 号寄存器。下图为使用串口猎人工具的高级发码功能，将需要设置的寄存器的值预先在发码列表中填入好，然后点击启动自动发码按钮，就能将这些数据帧自动下发。就能实现以 200K 的采样速率，对 1 个通道进行采样，共采集 16384 个数据。注意将循环次数设置为 1，不然会一直循环发送，四个寄存器对应的代码如下：

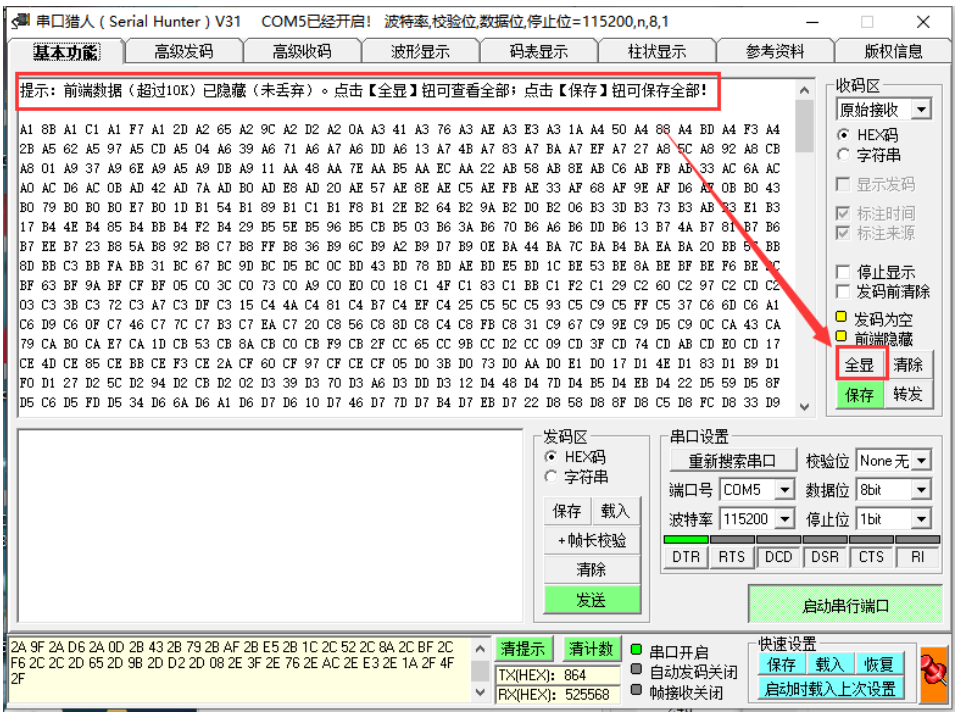
DataNum：　55 A5 02 00 00 40 00 F0

ChannelSel：　55 A5 01 00 00 00 01 F0

ADC_Speed_Set：　55 A5 03 00 00 00 F9 F0

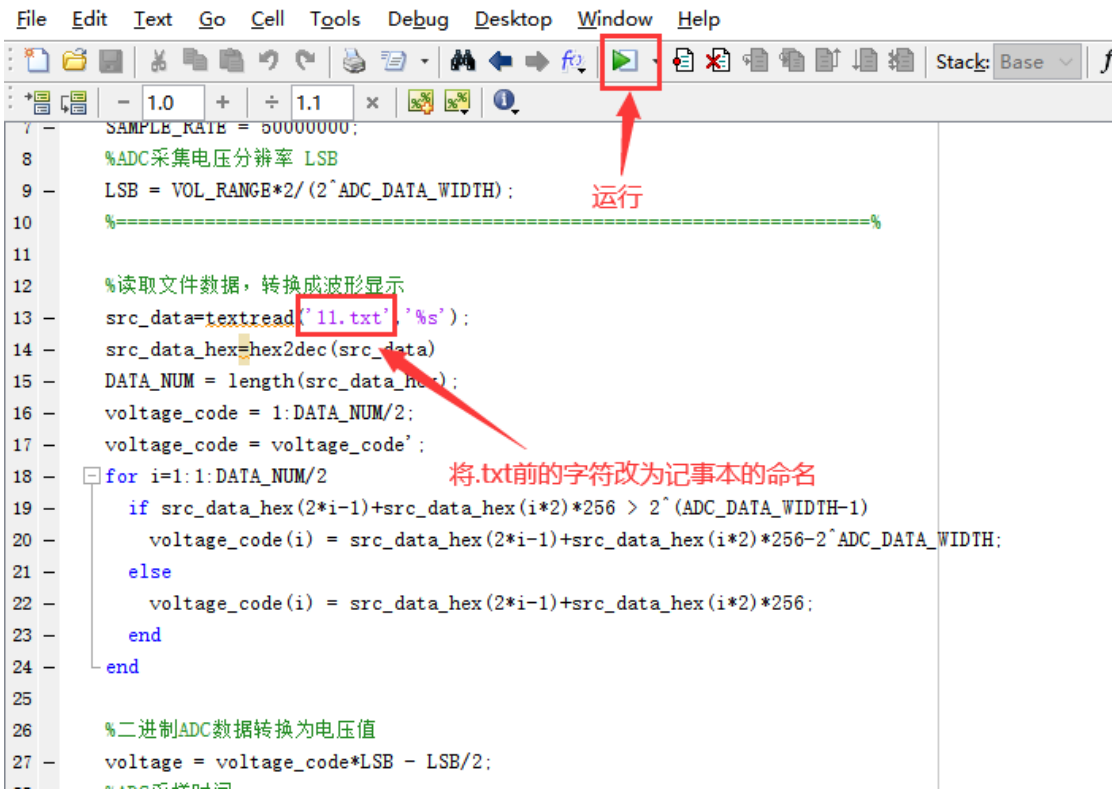
RestartReq：　55 A5 00 00 00 00 00 F0

下图则是发送完指令后 FPGA 传回的数据，注意这里由于数据过多而被隐藏，我们需要点击全显，让其全部显示出来，由于数据过多，为了能够直观的体现数据的完整性，我们将其数据拷贝到记事本中，通过 MATLAB 将数据转换为图像。

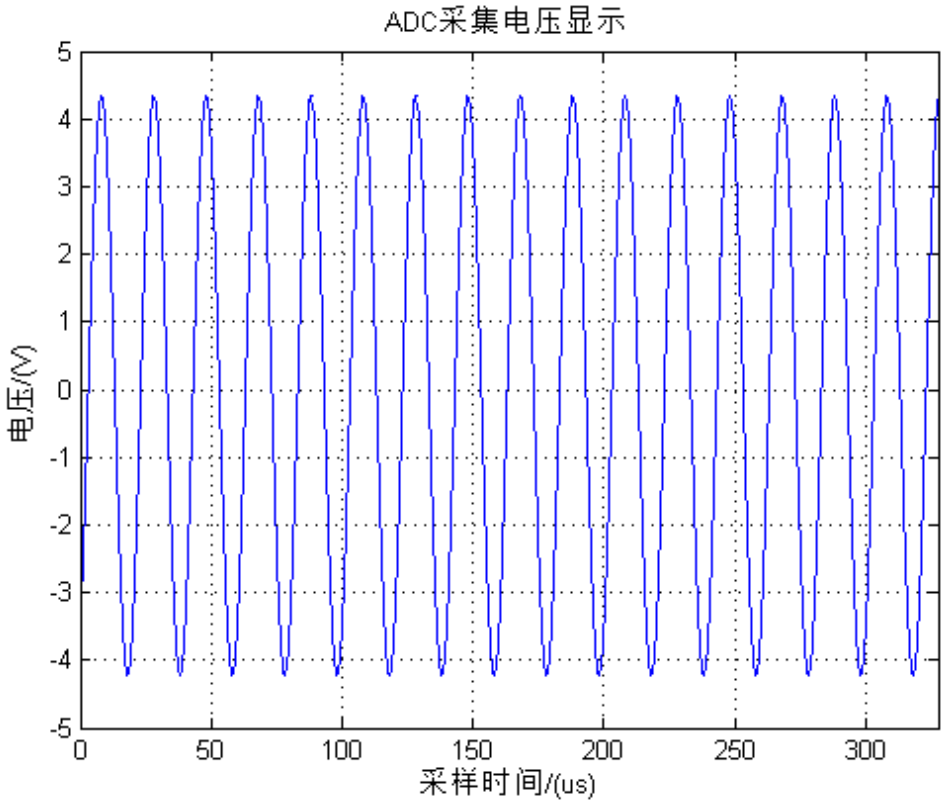


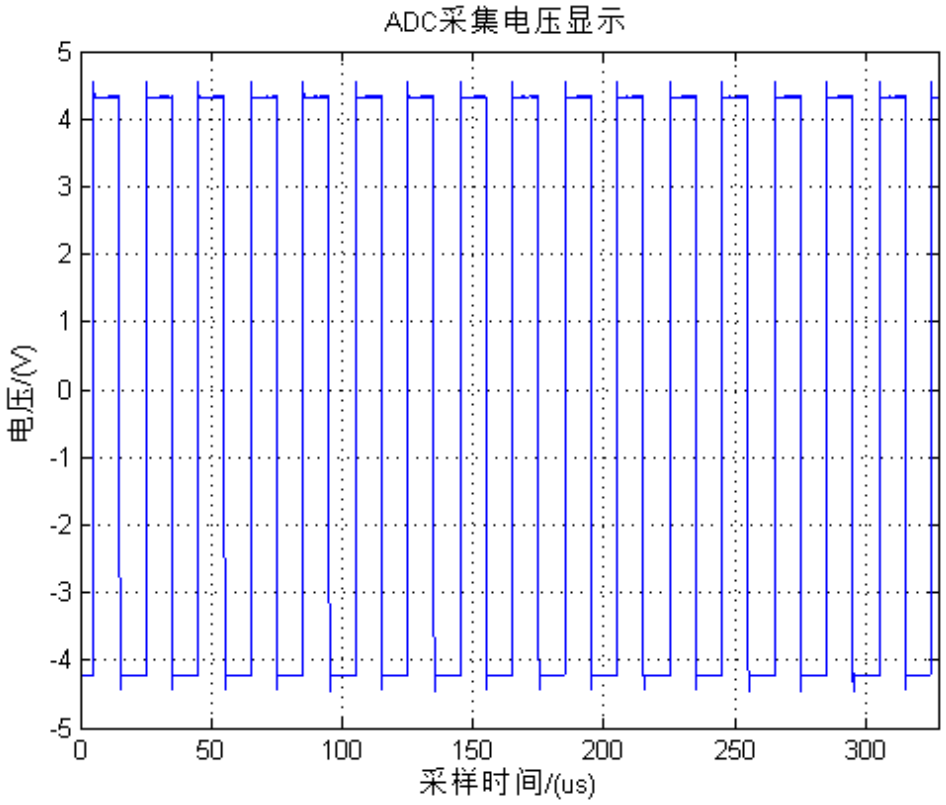
MATLAB 图像绘制

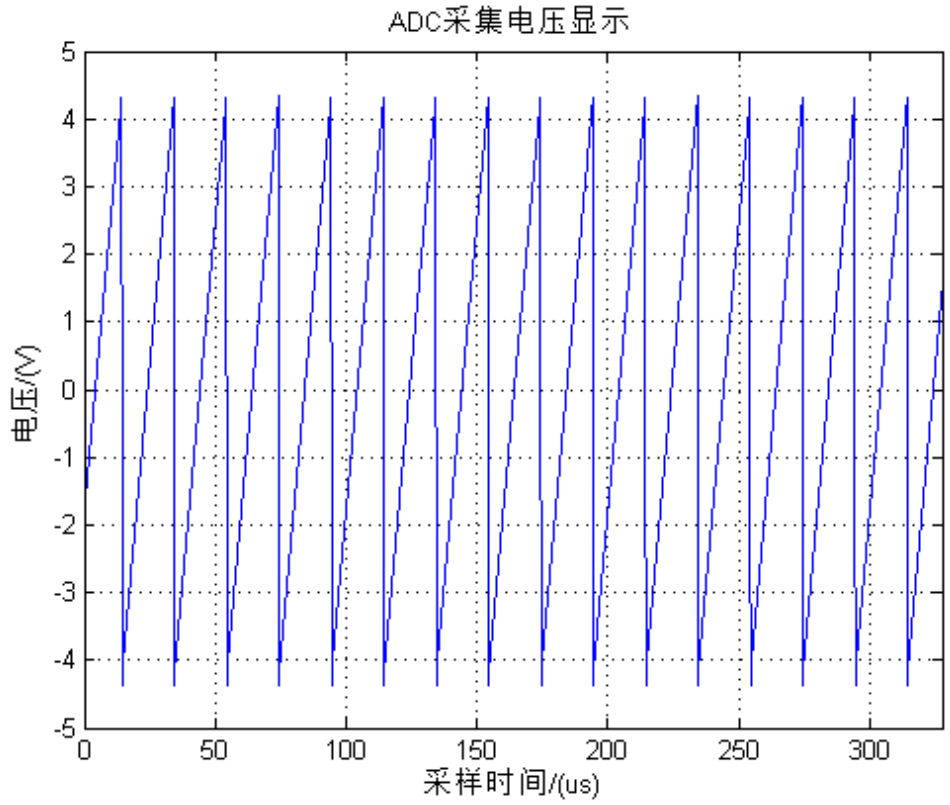
将拷贝好的记事本移动到 MATLAB 解压文件夹中（记得必须是纯英文路径），双击我们提供的 ADCdata_to_wave_v1_1.m 文件，在打开方式里选择以 MATLAB 打开，将方框中.txt 前的字符修改为你对记事本的命名，随后运行便可以直观的看到数据是否正确。



下面附上分别对正弦波，方波，三角波进行采样绘图最后得到的图形，相应的数据存放在本次实验提供的压缩包的 data 文件夹下。


ADC采集电压显示
电压/(V)
采样时间/(us)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
50
100
150
200
250
300


ADC采集电压显示
电压/(V)
采样时间/(us)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
50
100
150
200
250
300



总结

本次实验介绍了基于 AD7606 的串口收发，用户通过配置 AD7606 的四个寄存器，控制 ADC 进行采样，并将数据存储到 FIFO 中，再经由串口发送至电脑，借由串口调试工具对数据进行查看，通过相关的指令代码经由串口发送至 FPGA 中再配置 ADC 进行下一次采样，注意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。本次实验为以串口收发为基础的拓展实验，学有余力的同学可以尝试在本次实验的基础上进行修改，通过网口实现对采集到的数据进行传输以及指令的下发。

修订记录

V1.0	2021/02/28	首次发布