1 线型序列机与串行接口 ADC(TLC0832)驱动设计与验证

章节导读

模数转换器即 A/D 转换器，或简称 ADC（Analog to Digital Conver），通常是指一个将模拟信号转变为数字信号的电子元件。通常的模数转换器是把经过与标准量比较处理后的模拟量转换成以二进制数值表示的离散信号的转换器。

本章节以 TLC0832 为例介绍 AC101 开发板上 ADC 的工作原理及时序图解释，并用线性序列机（LSM）来描述时序图，进而正确驱动此类设备。本实验中，直接使用 AD/DA 模块上的 DAC 模块的输出直接接到 ADC 模块的输入上，通过控制 DAC 输出不同电压值，然后比较 DAC 的输出与 ADC 采样到的值的大小，来评估 ADC 驱动的正确性。

1.1 ADC 芯片概述及电路设计

在 AC101 FPGA 开发板上，使用了一片 8 位 2 通道的 DAC 芯片 TLV5618 来实现模拟到数字电压的转换，如下图所示。

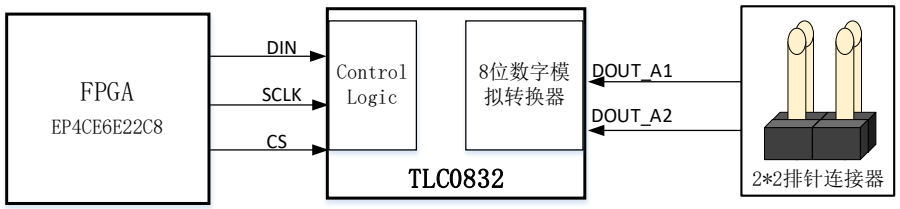


图 1 TLC0832 功能图

TLC0832 芯片内部结构如图 2 所示。该芯片是一个通过 SPI 接口发送 2 位指令帧到 TLC0832。通过其模拟输入端口接收模拟信号（通常是电压信号）。该信号在转换过程中会被采样并转换为数字信号。换完成后，TLC0832 通过 DOUT 引脚将 8 位的数字数据串行输出。输出的数据格式符合 SPI 协议，可以被微控制器或其他数字电路接收。

通过将 CS 设置为低来启动转换，使能所有逻辑电路。在整个转换过程中，CS 必须保持在低位。然后从处理器接收时钟输入。自动插入一个时钟周期的间隔，以允许所选的多路复用通道安定下来。SAR 比较器将电阻梯的连续输出与输入的模拟信号进行比较。比较器输出指示模拟输入是否大于或小于电阻梯输出。

当 CS 变高时，所有内部寄存器被清除。此时，输出电路进入高阻抗状态。

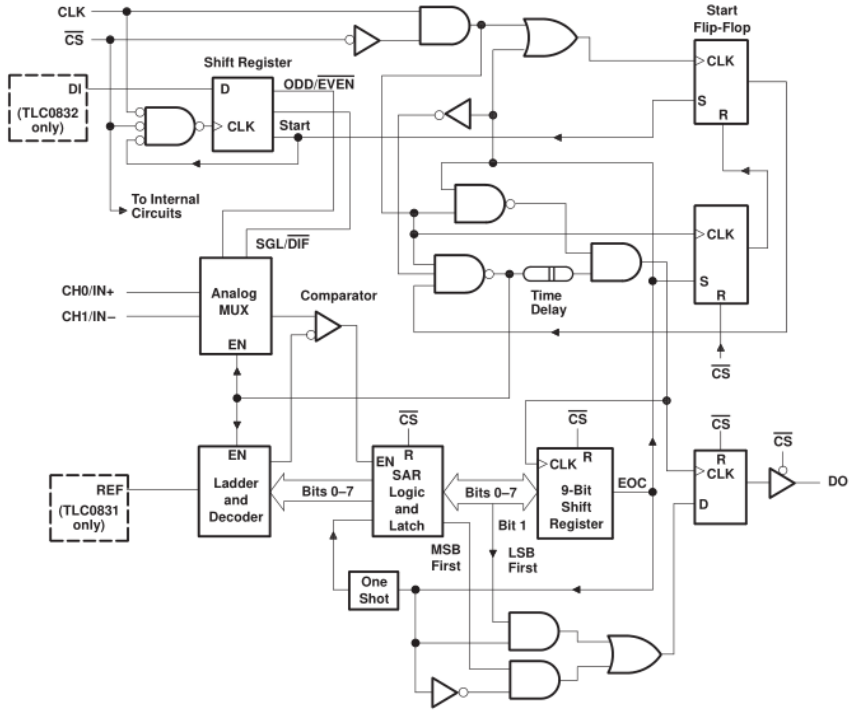


图 2 TLC0832 内部框图

1.2 TLC0832 工作电路设计

TLC0832 工作电路的部分电路图如下图所示：

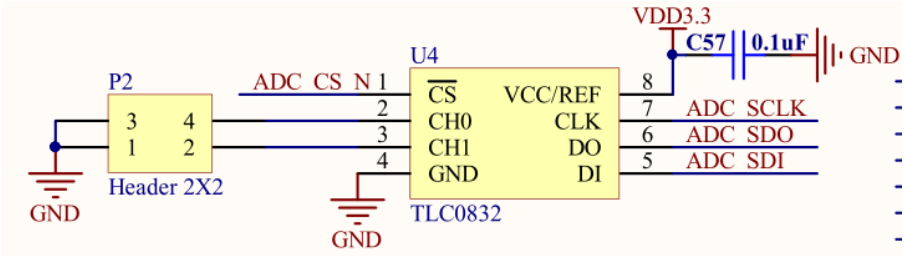


图 3 MCP4802 电路图

1.3 TLC0832 接口时序

当片选(CS)信号为低电平时，数据在每个 SCLK 信号的上升沿被采样。在第一个 SCLK 上升沿， 输入端的第一个逻辑高电平是起始位。TLC0832 的起始位后面有一个 2 位赋值字。在时钟输入的每个连续的上升沿中，起始位和赋值字将通过移位寄存器进行移位。当起始位被移到多路复用器寄存器的起始位置时，输入通道被选择并且转换开始。在转换期间，TLC0832 DI 端到多路复用器移位寄

存器被禁用。

数据通过 DOUT 端以串行方式输出，在 SCLK 的时钟作用下，从 MSB（最高位） 开始逐位输出数据。第一位为固定一周期的稳定时间，后八位为 ADC 转换结果数据。随后 CS 拉高。如图 4 所示。

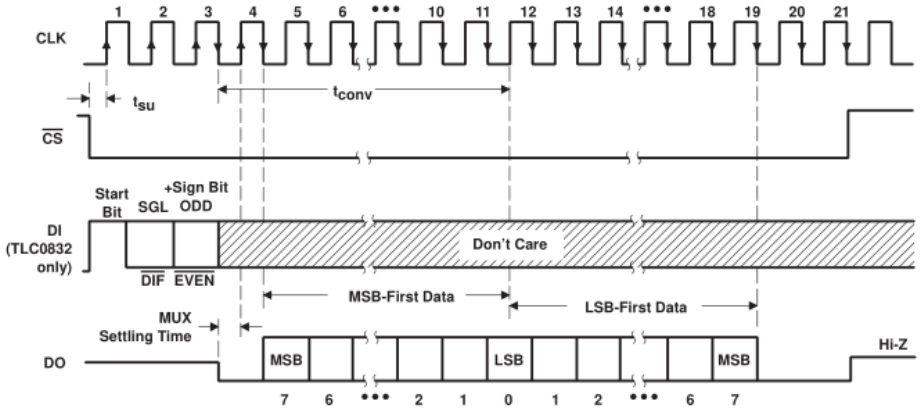


图 4 接口时序

1.4 TLC0832 写命令参数

TLC0832 通过 2 位指令控制通道选择以及差分。

表 1 控制指令

地址		通道	
SGL	ODD	CH0	CH1
L	L	+	-
L	H	-	+
H	L	+	
H	H		+

SGL 控制单端或者差分模式，ODD 控制通道选择。

TLC0832 MUX-ADDRESS CONTROL LOGIC TABLE

MUX ADDRESS		CHANNEL NUMBER	
SGL/$\overline{DIF}$	ODD/$\overline{EVEN}$	CH0	CH1
L	L	+	−
L	H	−	+
H	L	+	
H	H		+

H = high level, L = low level,
− or + = terminal polarity for the selected input channel

图 5 写命令

1.5 基于线型序列机的 DAC 驱动设计

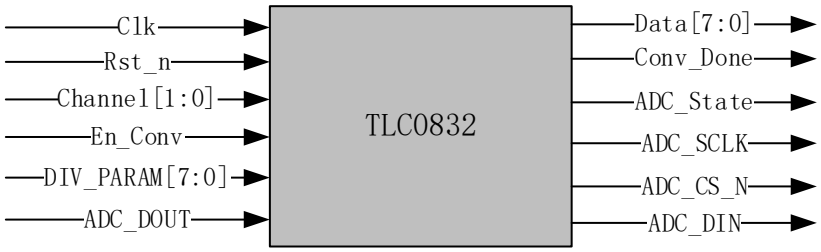


图 6 模块接口示意图

其中，每个端口的功能描述如表 2 所示。

表 2 MCP4802_DAC 模块端口功能描述

端口名称	I/O	端口功能描述
Clk	I	为控制器的工作时钟，频率为 50MHz，
Rst_n	I	控制器复位，低电平复位
Channel[1:0]	I	ADC 通道与差分控制
En_Conv	I	使能单次转换，高脉冲使能一次转换
DIV_PARAM	I	时钟分频设置，实际 SCLK 时钟　频率 = fclk /（DIV_PARAM * 2）
ADC_DOUT	I	ADC 转换结果，由 ADC 输给 FPGA

Data	O	ADC 转换结果
Conv_Done	O	转换完成信号，完成转换后产生一个时钟周期的高脉冲
ADC_State	O	ADC 工作状态，ADC 处于转换时为低电平，空闲时为高电平
ADC_SCLK	O	ADC 串行数据接口时钟信号
ADC_CS_N	O	ADC 串行数据接口使能信号
ADC_DIN	O	ADC 控制信号输出，由 FPGA 发送通道控制字给 ADC

在每个使能转换的时候，寄存 Channel 的值，防止在转换过程中该值发生变化。

```
always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)
        r_Channel <= 3'd0;
    else if(En_Conv)
        r_Channel <= Channel;
    else

        r_Channel <= r_Channel;
```

生成使能信号，当输入使能信号有效后便将使能信号 en 置 1，当转换完成信号有效时

便将其重新置 0。

```
always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)
        en  <= 1'b0;
    else if(En_Conv)
        en  <= 1'b1;
    else if(Conv_Done)
        en  <= 1'b0;
    else

            en  <= en;              DIV_CNT   <= DIV_CNT + 1'b1;

         end else

             DIV_CNT   <= 4'd0;
```

为了方便适配不同的频率需求率，设置了一个可调的计数器。根据计数器的

值周期性的产生 SCLK 时钟信号，这里可以将计数器的值等倍数放大，形成过采样。这里产生一个两倍于 SCLK 的信号，命名为 SCLK2X。

首先编写用于生成时钟 SCLK2X 的分频计数器。

```
always@(posedge Clk or negedge Rst_n)
if(!Rst_n)
    DIV_CNT  <= 8'd0;
else if(en)begin
    if(DIV_CNT == (DIV_PARAM - 1'b1))
        DIV_CNT  <= 8'd0;
    else
        DIV_CNT  <= DIV_CNT + 1'b1;
end else

        DIV_CNT  <= 8'd0;
```

根据使能信号以及计数器状态生成 SCLK2X 时钟。

```
always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)
        SCLK2X  <= 1'b0;
    else if(en && (DIV_CNT == (DIV_PARAM - 1'b1)))
        SCLK2X  <= 1'b1;
    else
        SCLK2X  <= 1'b0;
always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)

        SCLK2X  <= 1'b0;
```

每当使能转换后，对 SCLK2X 时钟进行计数。

```
 always@(posedge Clk or negedge Rst_n)

    if(!Rst_n)
        SCLK_GEN_CNT  <= 6'd0;
    else if(SCLK2X && en)begin
        if(SCLK_GEN_CNT == 6'd25)
            SCLK_GEN_CNT  <= 6'd0;
        else
            SCLK_GEN_CNT  <= SCLK_GEN_CNT + 1'd1;
    end else
        SCLK_GEN_CNT  <= SCLK_GEN_CNT;
```

根据 SCLK2X 计数器的值来确认工作状态以及数据传输进程。使用序列机

实现 ADC 串行数据接口的数据发送和接收。每个上升沿，使用循环移位寄存器来寄存芯片数据输出线上的转换结果。一次转换过程中总共循环 8 次。

```
        always@(posedge Clk or negedge Rst_n)

    if(!Rst_n)begin
        ADC_SCLK <= 1'b1;
        ADC_CS_N <= 1'b1;
        ADC_DIN  <= 1'b0;
    end else if(en) begin
        if(SCLK2X)begin
            case(SCLK_GEN_CNT)
                6'd0:begin ADC_CS_N <= 1'b0; end
                6'd1:begin ADC_SCLK <= 1'b0; ADC_DIN  <= 1'b1; end
                6'd2:begin ADC_SCLK <= 1'b1; end
                6'd3:begin ADC_SCLK <= 1'b0; ADC_DIN  <=
r_Channel[1];end   //addr[1]
                6'd4:begin ADC_SCLK <= 1'b1; end
                6'd5:begin ADC_SCLK <= 1'b0; ADC_DIN  <=
r_Channel[0];end   //addr[0]
                6'd6:begin ADC_SCLK <= 1'b1; end
                6'd7:begin ADC_SCLK <= 1'b0; ADC_DIN <= 1'b0; end
                6'd8:begin ADC_SCLK <= 1'b1; end

                //每个上升沿，寄存 ADC 串行数据输出线上的转换结果
                6'd10,6'd12,6'd14,6'd16,6'd18,6'd20,6'd22,6'd24:
                    begin ADC_SCLK <= 1'b1; r_data <= {r_data[6:0],
ADC_DOUT}; end  //循环移位寄存 DOUT 上的 8 个数据

                6'd9,6'd11,6'd13,6'd15,6'd17,6'd19,6'd21,6'd23:
                    begin ADC_SCLK <= 1'b0; end

                6'd25:begin ADC_CS_N <= 1'b1; end
                default:begin ADC_CS_N <= 1'b1; end //将转换结果输出
            endcase
        end
        else ;
    end else begin
        ADC_CS_N <= 1'b1;
    end
```

一次转换结束的标志，即 en && SCLK2X && (SCLK_GEN_CNT == 6'd33) 为真，并产生一个高脉冲的转换完成标志信号 Conv_Done。一次转换结束后将内部寄存器 r_data 的数据输出到输出端 Data 上。

ADC 工作状态，在手册中看出 CS_N 在工作时为低电平，因此直接将此信号作为工作状态指示。

```
 always@(posedge Clk or negedge Rst_n)

if(!Rst_n)begin
    Data <= 8'd0;
    Conv_Done <= 1'b0;
end else if(en && SCLK2X && (SCLK_GEN_CNT == 6'd25))begin
    Data <= r_data;
    Conv_Done <= 1'b1;
end else begin
    Data <= Data;
    Conv_Done <= 1'b0;
end

//产生 ADC 工作状态指示信号
assign ADC_State = ADC_CS_N;
```

完成工程代码设计后，接下来进行激励创建及仿真测试。

1.6 激励创建及仿真测试

为了测试模块功能，模拟 ADC 芯片的输出。这里用 多功能 mif 生成器 产生一个正弦波文件，位宽 8 位，进制 16 位，个数为 2048，并以 sine_8bit.txt 保存当前工程下的 simulation 目录下的 modelsim 文件夹。

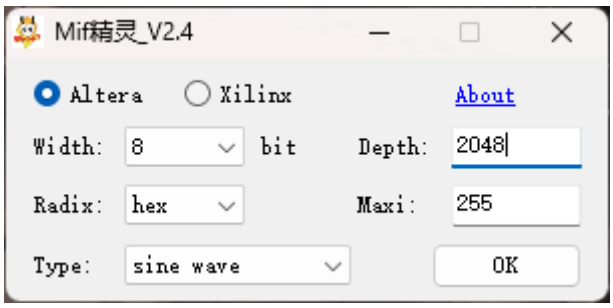


图 7 Mif 精灵设置

【小软件合集】FPGA 课程教程中常用软件和小工具合集

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

(出处: 芯路恒电子技术论坛)

这样就需要将产生的数据，发送到 ADC 驱动模块的输入线 DOUT 上，这

里使用系统任务$readmemb，其可以用来从文件中读取数据到存储器中。其格式有以下三种：

$readmemb("<数据文件名>",<存贮器名>);

$readmemb("<数据文件名>",<存贮器名>,<起始地址>);

$readmemb("<数据文件名>",<存贮器名>,<起始地址>,<结束地址>。

首先定义文件存放位置。

```
`define sin_data_file "./sine_8bit.txt"
```

定义 2048 个 8 位的存储单元，然后将波形数据读取到存储器中。

```
reg[7:0]  memory[2047:0];//测试波形数据存储空间
initial $readmemh(`sin_data_file,memory);//读取原始波形数据读到 memory
中
```

在测试时将这些数据整体循环三次，进行验证。其中 gene_DOUT(memory[address])；依次将存储器中存储的波形数据读出，按照 ADC 芯片转换结果输出方式的送到 DOUT 信号线上。

```
    initial begin
        Rst_n = 0;
        Channel = 0;
        En_Conv = 0;
        ADC_DOUT = 0;
        address = 0;
        #101;
        Rst_n = 1;
        #100;
        Channel = 3;
        for(i=0;i<3;i=i+1)begin
            for(address=0;address<2047;address=address+1)begin
                En_Conv = 1;
                #20;
                En_Conv = 0;
                gene_DOUT(memory[address]); //依次将存储器中存储的波形读
出，按照 ADC 的转换结果输出方式送到 DOUT 信号线上
                @(posedge Conv_Done);   //等待转换完成信号
                #200;
            end
        end
        #20000;
```

```
        $stop;
    end
```

将存储空间的数据按照 ADC 的数据输出格式，发送到 DOUT 信号线上，供控制模块采集。

```
    task gene_DOUT;
        input [11:0]vdata;
        reg [4:0]cnt;
        begin
            cnt = 0;
            wait(!ADC_CS_N);
            while(cnt<12)begin
                @(negedge ADC_SCLK) ADC_DOUT = vdata[11-cnt];
                cnt = cnt + 1'b1;
            end
        end
    endtask
```

ADC 的输出结果只有 8 位，这里产生模拟的 ADC 采样结果时，将 Vdata 的位宽定义为了 12 位，这是因为 TLC0832_ADC 一次传输是 12 个 SCLK，这里从第一个 SCLK 就开始产生数据位，而不是等到计数过 4 位之后再传真实的数据。如果输入的 vdata 是 8 位，就需要在这段代码中数 4 个 SCLK 之后再开始一位一位的输出数据了。为了简化代码，将位宽定义为 12 位。

这样，多出来的最开始 4 位数据在 task 被调用的时候，传入的是一个 8 位的数，所以默认高 4 位会是 0，因此不会对转换结果有任何影响。开始仿真后，可看出手动控制 DAC_DATA 数据输入状态正常，如下图**错误!未找到引用源。**所示。

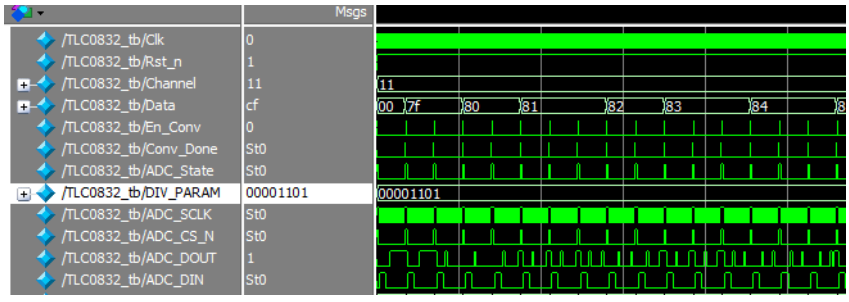


图 8 整体仿真波形

放大第一个数据传输过程，如下图所示，可看出此次信号较多。这里首先查看与 ADC 相关的使能信号、状态标志信号以及 SCLK 时钟信号等。选中以上这些信号，可以看出，每当外界输入一个周期的 En_Conv 转换信号，模块内部

使能信号 en 就置高，一直到转换完成标志信号 Conv_Done 有效再置 0。当 SLK2X 计数器值为 25d 且当 SCLK2X 为高时，便输出一个时钟周期的高脉冲信号，标志着本次转换过程结束。同时 SCLK2X 时钟为 ADC 工作时钟 SCLK 的两倍。以上各信号符合既定的设计。

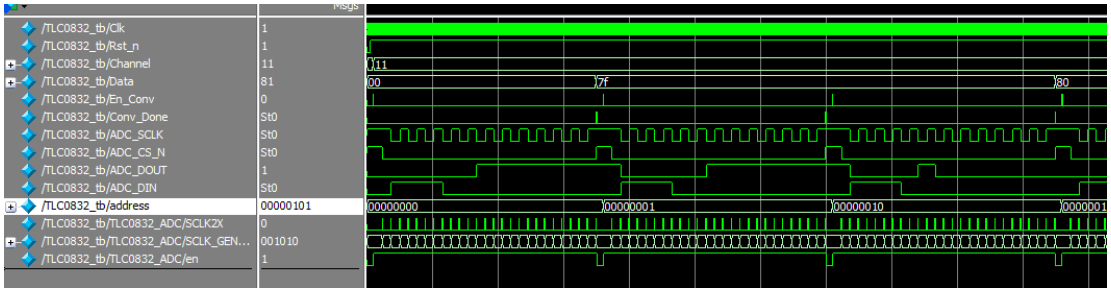


图 9 第一次传输数据仿真波形

将并行输出数据 Data 设置为模拟形式，鼠标右键点击 Data 信号，在弹出的窗口中选择 format—>Analog(Custom)，然后在弹出的参数设置框 Format 选项卡中，设置模拟信号最大值为 255，最小值为 0，也就是 8 位数据能够表示的数据范围。波形所占高度可根据实际情况自行设计，此处暂定 100。如下图所示。

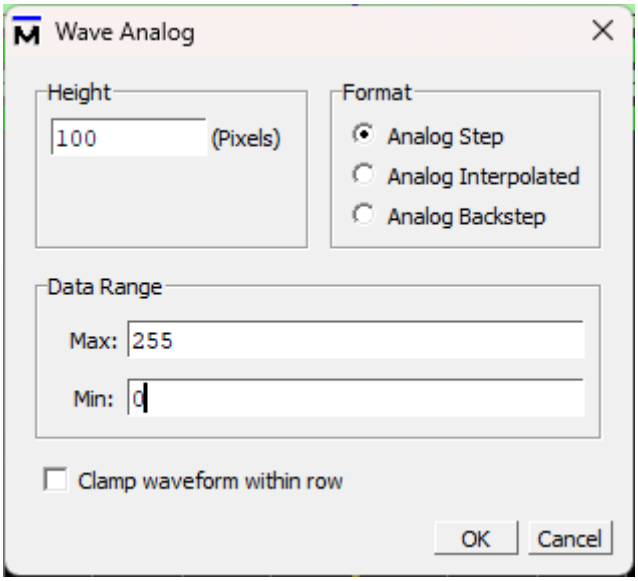


图 10 波形设置

重启仿真后可看出数据输入的正弦波也就是采样正确，符合既定设计。

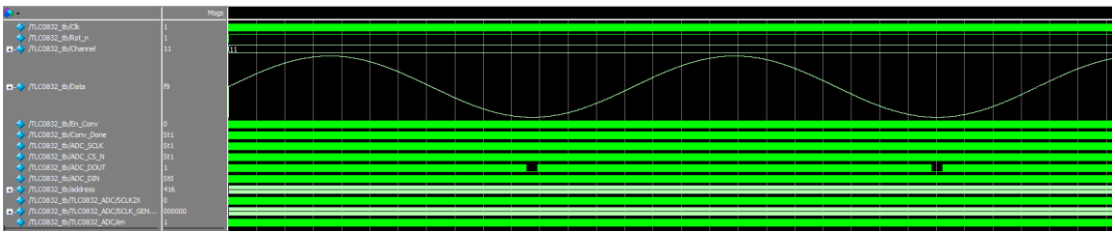


图 11 采样数据波形

通过以上波形观察可以验证 TLC0832_ADC 转换芯片驱动设计的正确性。完成仿真验证后，接下来将进行 ISSP 配置，进而实现板级验证。

1.7 ISSP 配置及板级验证

经过以上设计和仿真分析，本工程设计的 TLC0832 的驱动模块功能已经实现。接下来就可以使用该控制器来控制开发板上的 TLC0832 型芯片。

首先，完成工程的管脚绑定，对应关系如下：

信号名	方向	AC101
Clk	Input	PIN_23
Rst_n	Input	PIN_24
DAC_DIN	Output	PIN_98
DAC_SCLK	Output	PIN_99
DAC_LDAC	Output	PIN_87
DAC_SCLK	Output	PIN_99
DAC_CS_N	Output	PIN_100
ADC_CS_N	Output	PIN_86
ADC_DIN	Output	PIN_104
ADC_DOUT	Input	23

ADC_SCLK	Output	101

接下来就可以使用该控制器来控制开发板上的 TLC0832 型 ADC 芯片。作为模数转换芯片，其功能是将模拟电压转为数字信号，因此，测试时需要有被测试的模拟电压信号。这里可以借助前一节设计的基于 MCP4802 的数模转换系统，来输出指定的电压信号，并将该信号接入 TLC0832 芯片的模拟输入端口。再由 FPGA 通过本节设计的 TLC0832 控制器控制 TLC0832 芯片采集转换该电压信号。并将采样结果和 MCP4802 型 DAC 的输出设定电压进行比较，如果两者值一致，则可以确定本节设计的 TLC0832 控制器能够实现对该芯片的转换控制和读取功能。

对于 MCP4802 输出的电压值，可以继续采用前一节使用的 ISSP 功能来控制和调整。而对于 TLC0832 采样通道，也可以使用 ISSP 的 Source 功能来设置，同时，使用 ISSP 的 Probe 功能来读取 TLC0832 控制器接收得到的模数转换结果。

整个测试系统如下图所示。

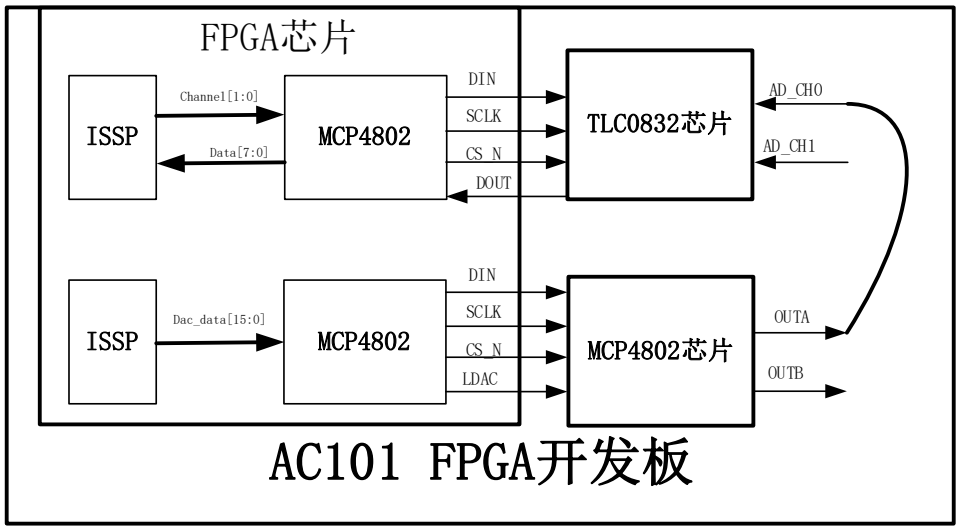


图 12 ISSP 源位宽参数设置

为了对 TLC0832 驱动模块进行板级验证，可以配置和使用 ISSP 在线调试工具进行数模转换的数据量设定。

首先创建一个 ISSP_DAC IP 核，其源位宽设定为 16 位。如下图所示：

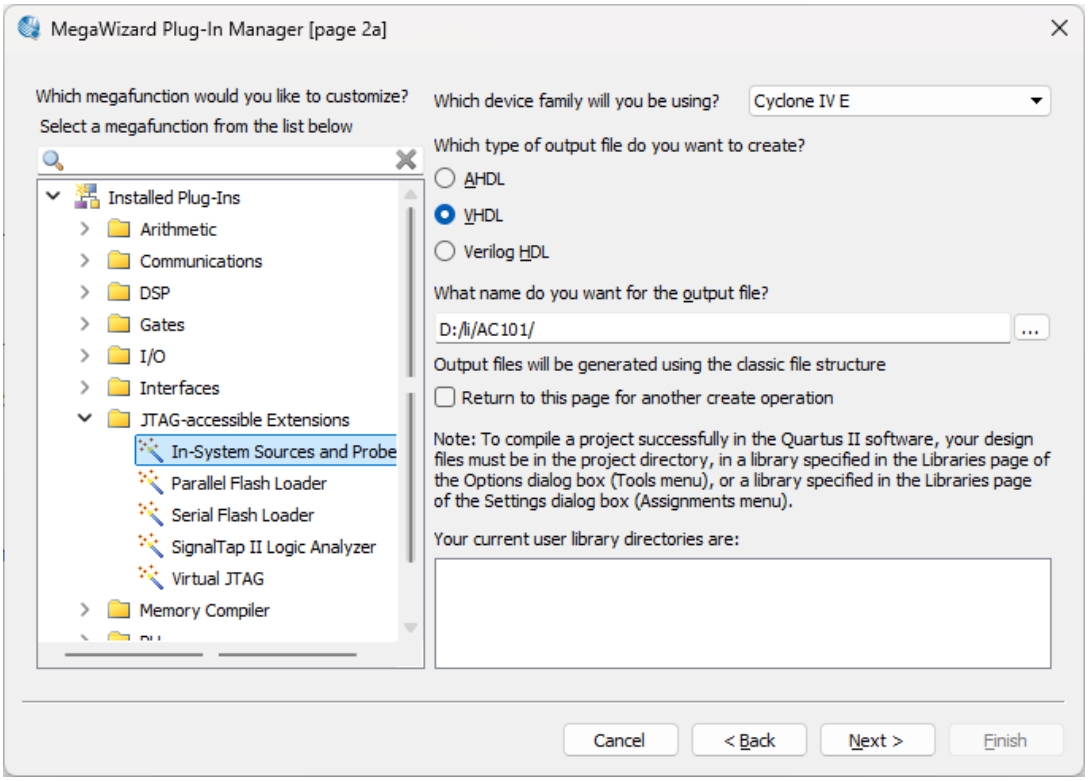


图 13 ISSP 创建

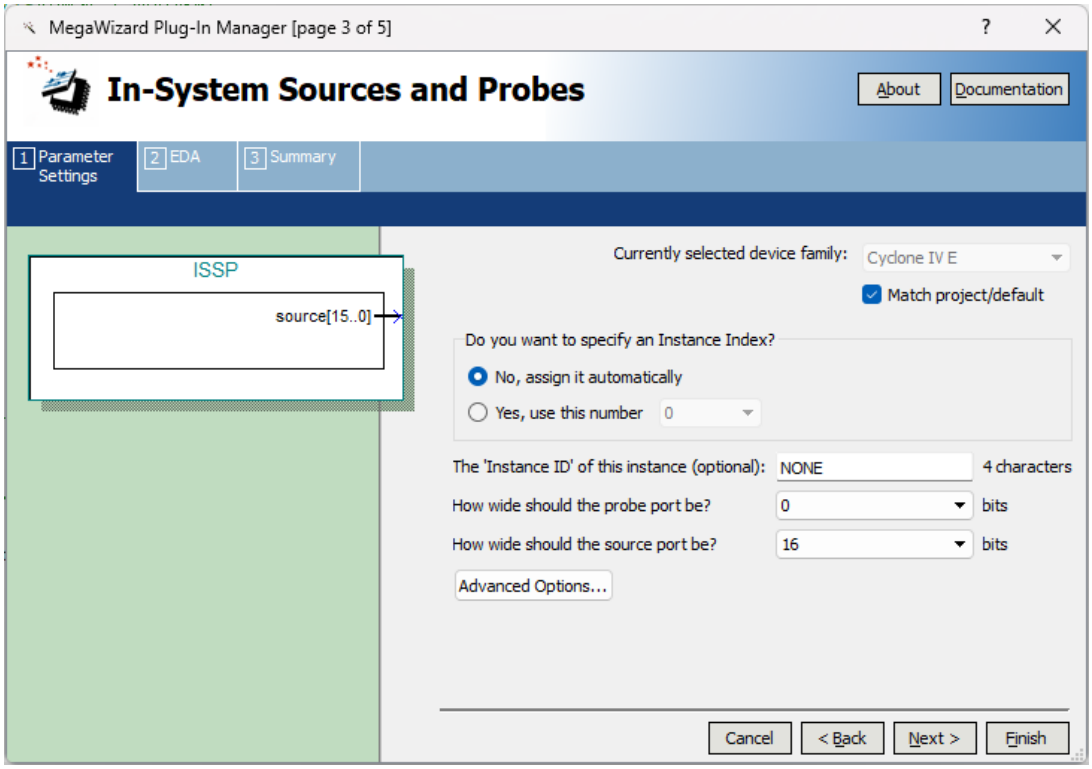


图 14 ISSP_DAC 参数设置

然后创建 ISSP_ADC IP 核，源位宽为 2，探针位宽为 8。

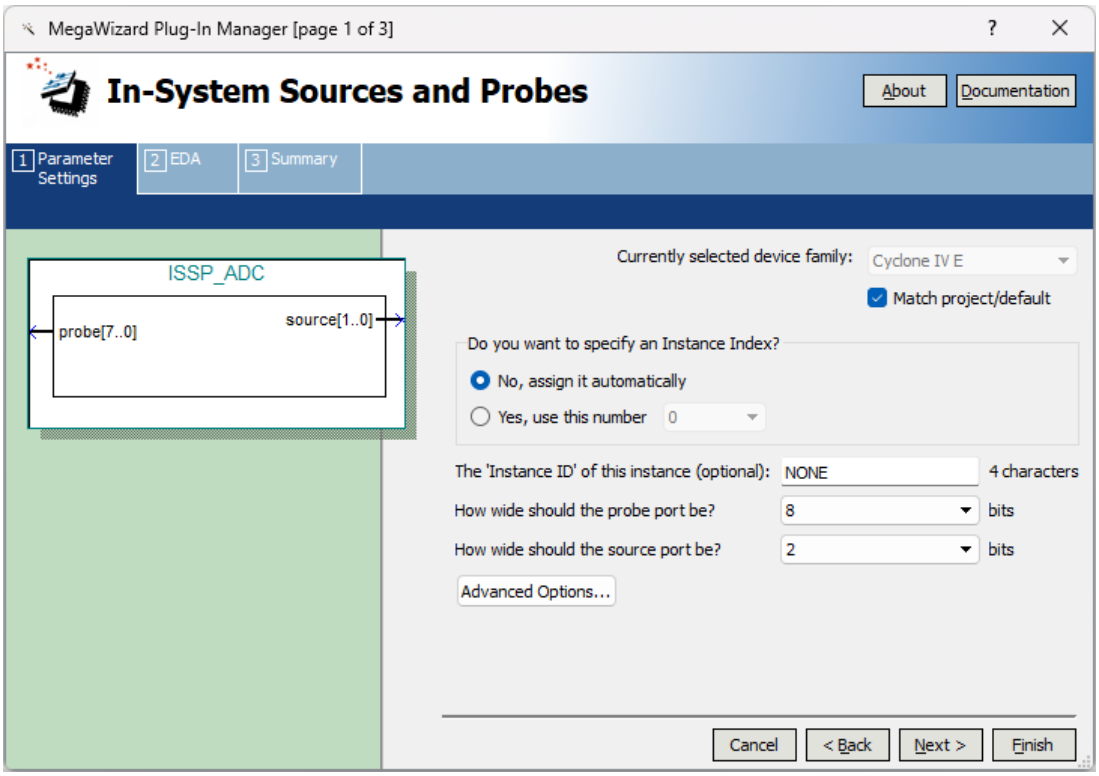


图 15ISSP_ADC 参数配置

加入工程后新建顶层文件 ADC_test.v，并对 ISSP 以及设计好的 MCP4802 和 TLC0832 驱动模块进行例化，如有需要，供参考的顶层文件可随工程查阅。

之后连接到开发板的电缆，将 ADC 芯片的 CH1 与 DAC 芯片的 CHA 用杜邦线连接。并下载程序。

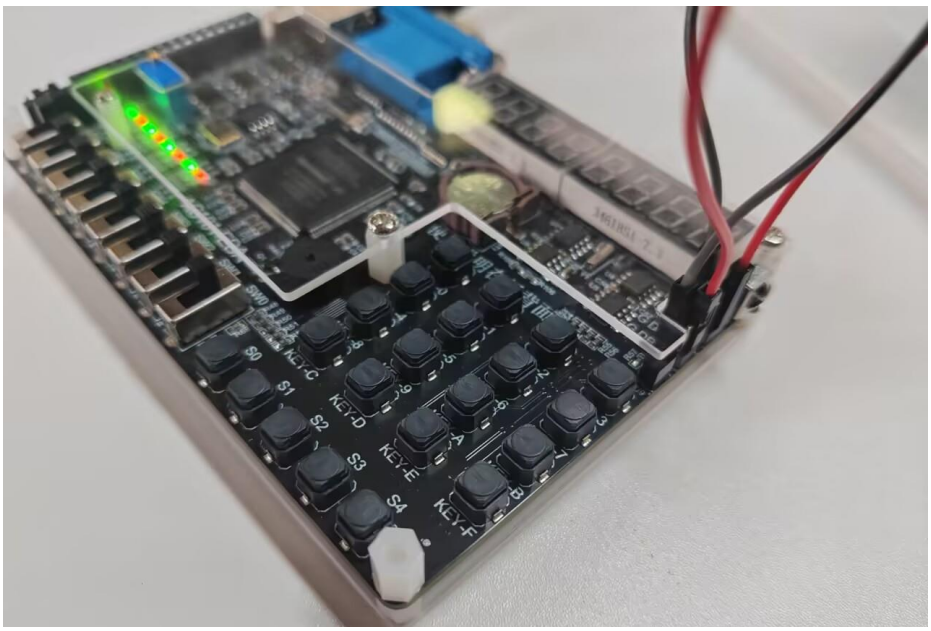
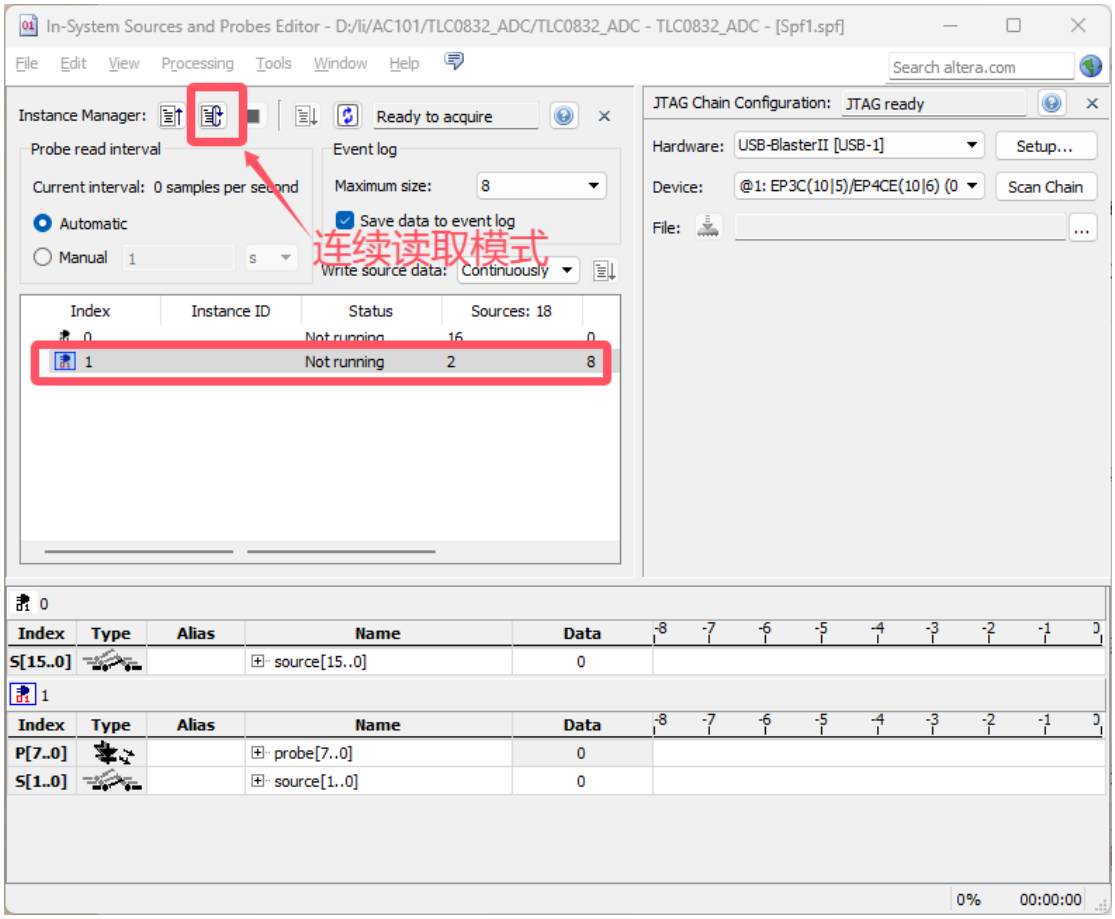


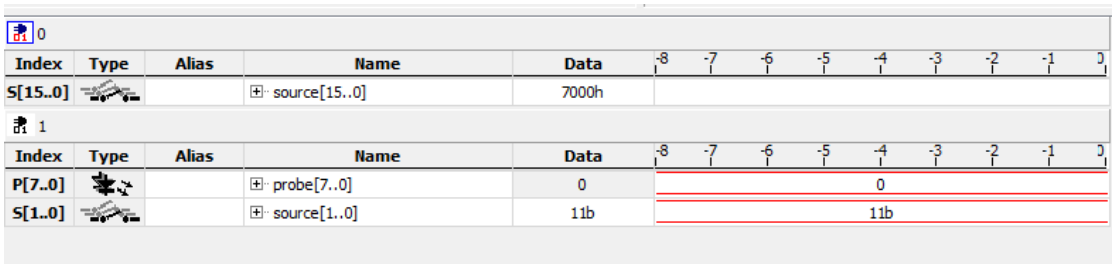
图 16 板级验证连接图

需要注意的是：图中模拟量输入输出端口靠下方的红色杜邦线接头为模拟量传导线缆，靠上方的灰色杜邦线接头为 GND。

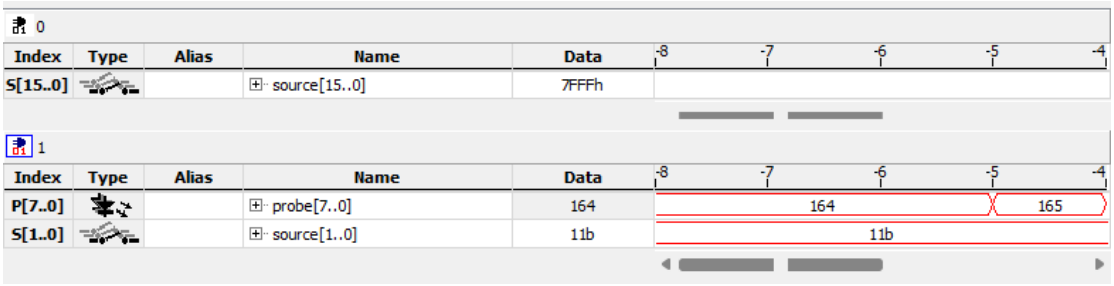
完成上述工作随后启动 ISSP，将 ADC 探针启动连续读取模式。接下来可以进行如下测试：



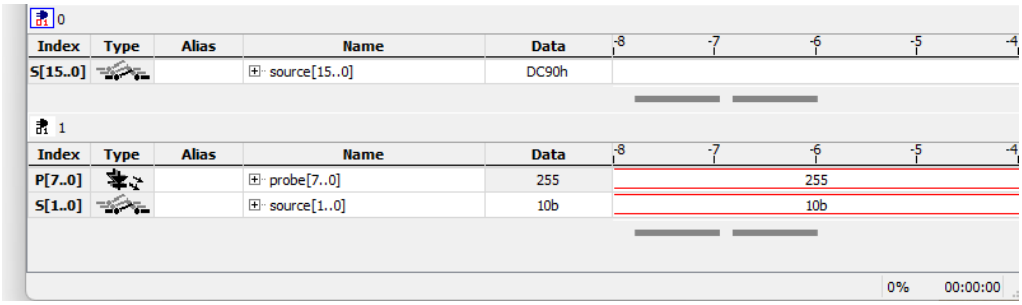
DAC 的模拟电压 A 通道输出端(AC101 开发板上标注丝印为 DA)连接 ADC 的 A1 输入端。如下图所示，使得 DAC 输出电压为 0，可测得电压为 0。



此时更新 DAC 的模拟电压 A 通道的输出值为 7FFFh，如图下图所示，可以看出 A1 测量数据为 164。其理论电压值为 2.048V，实际电压为 164/255*3.3=2.12V，在误差允许范围内。



同理 DAC 的模拟电压 B 通道输出端(AC101 开发板上标注丝印为 DB)连接 ADC 的 A0 输入端。如下图所示，可以看出 A0 测量数据为 164。其理论电压值为 3.216V，实际电压为 255/255*3.3=3.3V，在误差允许范围内。



1.8 常见问题说明

1、配置 ISSP 工具时，需注意区分当前配置的 ISSP 是 ADC 的数据源还是 DAC 的数据源，并进行有针对性的位宽配置。

2、模数转换的转换结果可以有两个输出通道进行选择。单独选择哪个通道或者是否为差分输出，需由寄存器配置决定。

3、TLC0832 的参考电压为 3.3V，计算是应按照 3.3V 计算。