

5.3 线性序列机与串行接口 ADC 驱动设计与验证

本节导读

模数转换器即 A/D 转换器，或简称 ADC（Analog to Digital Converter），通常是指一个将模拟信号转变为数字信号电子元件。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。

模数转换器的种类很多，按工作原理的不同，可分成间接 ADC 和直接 ADC。间接 ADC 是先将输入模拟电压转换成时间或频率，然后再把这些中间量转换成数字量，常用的有双积分型 ADC。直接 ADC 则直接转换成数字量，常用的有并联比较型 ADC 和逐次逼近型 ADC。

并联比较型 ADC：采用各量级同时并行比较，各位输出码也是同时并行产生，所以转换速度快。并联比较型 ADC 的缺点是成本高、功耗大。

逐次逼近型 ADC：它产生一系列比较电压 V_R ，但它是逐个产生比较电压，逐次与输入电压分别比较，以逐渐逼近的方式进行模数转换。它比并联比较型 ADC 的转换速度慢，比双积分型 ADC 要快得多，属于中速 ADC 器件。

双积分型 ADC：它先对输入采样电压和基准电压进行两次积分，获得与采样电压平均值成正比的时间间隔，同时用计数器对标准时钟脉冲计数。它的优点是抗干扰能力强，稳定性好；主要缺点是转换速度低。

本节以 ADC128S022 为例介绍 ADC 的工作原理及时序图解释，并用线性序列机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。

5.3.1 ADC 芯片概述及电路设计

1. ADC128S022 型 ADC 内部工作原理

在 AC620 开发板上使用的模数转换器为逐次逼近型的低功耗芯片 ADC128S022，其具有 8 通道以及 12 位的分辨率。电源采用独立的模拟供电以及数字供电，其中模拟电源 V_A 输入范围为 2.7V~5.25V，数字电源 V_D 输入范围为 2.7V~ V_A 。其与外部通信支持多种接口如：SPI、QSPI、MICROWIRE 以及通用的 DSP 接口。转换速度在 50kps~200kps，典型情况下当 3V 供电时功耗为 1.2mW，5V 供电时为 7.5mW。

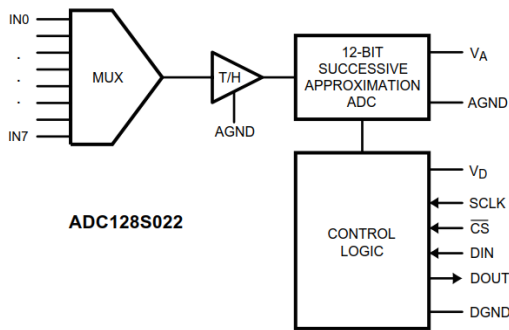


图 25.1 内部功能框图

本款 ADC 为 12 位分辨率，因此 1bit 代表的电压值即为 $V_A/4096$ 。手册中同时指出，当模拟输入电压低于 $V_A/8192$ 时，输出数据即为 0000_0000_0000b。同理由于芯片本身内部构造当输出数 0000_0000_0000b 变为 0000_0000_0001b 时，实际输入电压变化为 $V_A/8192$ 而不是 $V_A/4096$ 。当输入电压大于等于 $V_A - 1.5 \cdot V_A/4096$ ，输出数据即为 1111_1111_1111b。

2. ADC128S022 型 ADC 芯片引脚功能

ADC128S022 芯片引脚及功能描述如表 25.1 所示。

表 25.1 芯片引脚功能描述

引脚名	编号	I/O	功能描述
$\overline{CS}$	1	I	片选端，下降沿时开始测量转换，低电平状态为正在转换
V_A	2	供电	模拟电源输入正，也就是参考电压。2.7V~5.25V
AGND	3	供电	模拟地
IN0~IN7	4~11	I	模拟输入脚，电压输入范围为 $0 \sim V_{REF}$
DGND	12	供电	数字地
V_D	13	供电	数字电源输入正，2.7V~ V_A 。
DIN	14	I	串行数据输入脚，SCLK 上升沿采样输入
DOUT	15	O	转换结构输出脚，SCLK 下降沿输出
SCLK	16	I	时钟输入，频率范围为 0.8~3.2MHz。

3. ADC128S022 电路设计

ADC128S022 部分电路图如图 25.2 所示。这里外接了一个抗混叠低通滤波器，其作用是为了避免输入信号的高频成分在 ADC 的基带中引起混叠。数字电源与模拟电源电压均为 3.3V，且数字电源与模拟电源之间串联磁珠用于抑制电磁干扰。

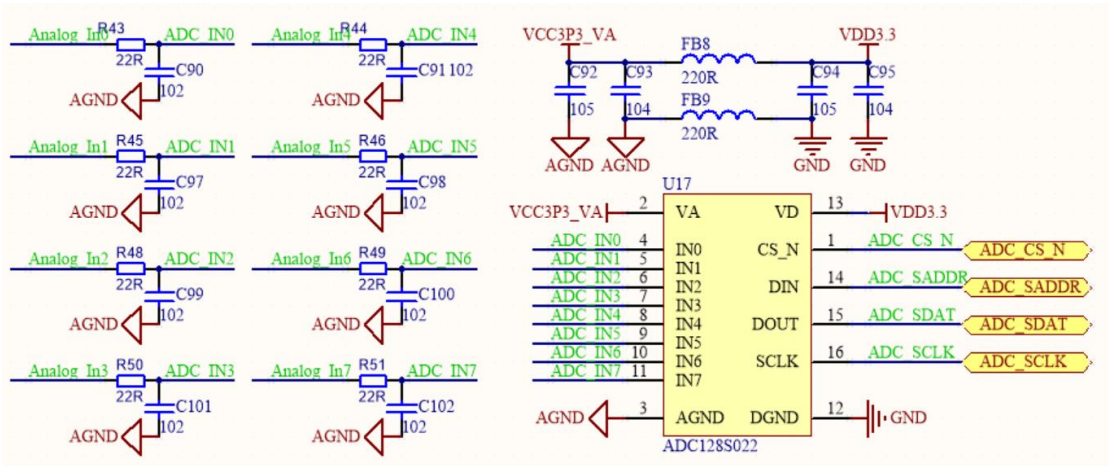


图 25.2 ADC128S022 部分电路图

5.3.2 ADC128S022 型 ADC 接口时序

微控制器与 ADC128S022 的接口如图 25.3 所示，该接口使用标准的 SPI 接口，因此可以直接连接到微控制器的片上 SPI。对于 FPGA 来说，则可按照 SPI 时序搭建控制电路，以实现

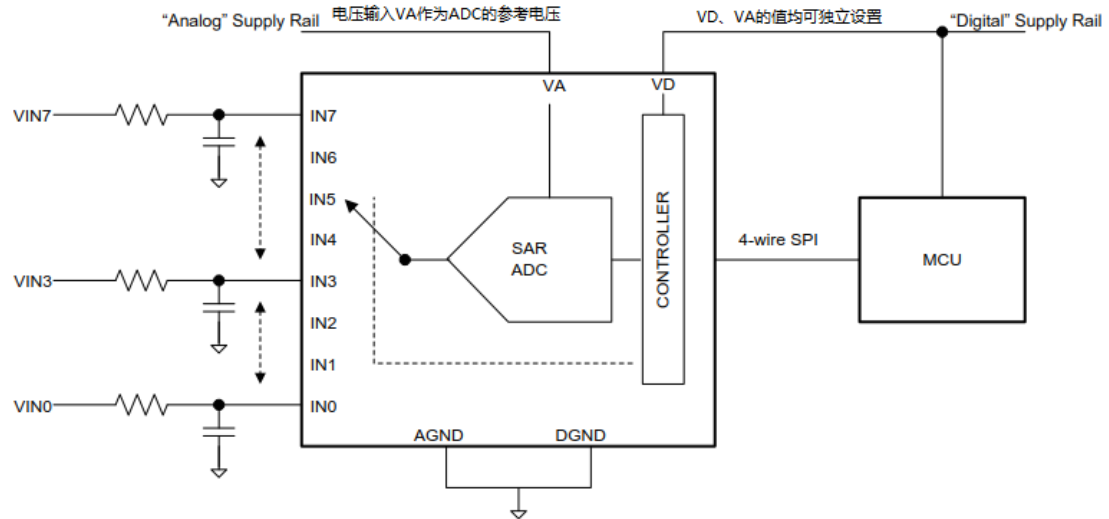


图 25.3 主机与 ADC128S022 接口示意图

ADC128S022 通过 SPI 接口与控制器进行通信的时序图如图 所示。一个串行帧开始于 $\overline{CS}$ 的下降沿，结束于 $\overline{CS}$ 的上升沿。一帧包含 16 个上升沿 SCLK。ADC 的 DOUT 引脚当 $\overline{CS}$ 为高时代表空闲状态，当为低时为传输状态。也就是相当于 $\overline{CS}$ 可以充当输出使能。在 $\overline{CS}$ 为高时 SCLK 默认高。在头三个 SCLK 的循环，ADC 处于采样模式（track）。在接下来的 13 个循环为保持模式（hold），转换完成并且完成数据输出。下降沿 1~4 为前导零，5~16 为输出转换结果。如果在持续测量模式中，ADC 在 $N \times 16$ 个 SCLK 的上升沿会自动进去采样模式，在 $N \times 16 + 4$ 个下降沿进入保持/转换模式。

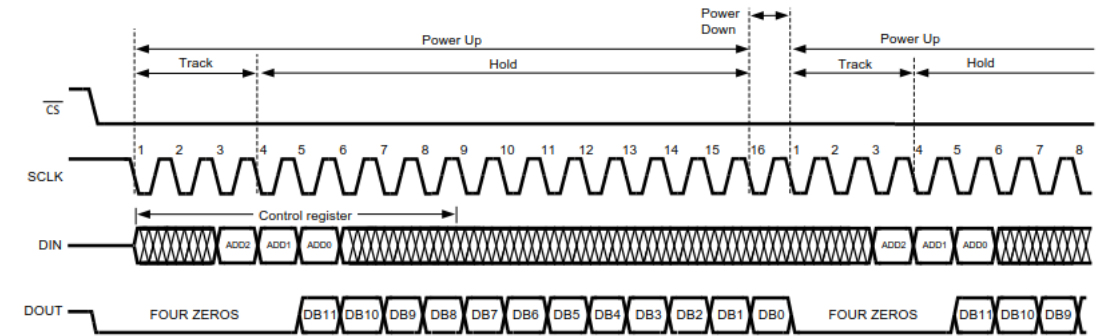


图 25.4 SPI 接口时序图

进入采样模式有三种方式，第一种当 SCLK 为高电平时 CS 变为低，当 SCLK 第一个下降沿到来时便进入采样模式，如时序图所示；第二种当 SCLK 为低电平时 $\overline{CS}$ 变低，自动进入采样模式， $\overline{CS}$ 的下降沿即视为 SCLK 的第一个下降沿；第三种 SCLK 与 $\overline{CS}$ 一同变为低，这样对于两信号上升沿没有时序约束，但对于其下降沿有要求。

在 DIN 的 8 个控制寄存器，每一位代表的含义如表 25.2 所示。

表 25.2 DIN 控制寄存器

Bit7(MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
-----------	------	------	------	------	------	------	------

DONTC	DONTC	ADD2	ADD1	ADD0	DONTC	DONTC	DONTC
-------	-------	------	------	------	-------	-------	-------

其中 ADD[2:0]代表的输入通道选择如表 25.3 所示。

表 25.3 ADD 寄存器与输入通道选择对应表

ADD2	ADD1	ADD0	输入通道
0	0	0	IN0 (Default)
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	IN5
1	1	0	IN6
1	1	1	IN7

5.3.3 ADC128S022 接口时序设计

经查阅手册可知器件工作频率 SCLK 推荐范围为 0.8~3.2MHz，这里定义其工作频率为 1.92MHz(周期为 520ns)。设置一个两倍于 SCLK 的采样时钟 SCLK2X，使用 50M 系统时钟十三分频而来即 SCLK2X 为 3.84MHz。针对 SCLK2X 进行计数来确定图 25.4 中各个信号的状态。结合前面 ADC 的接口时序图，按照线性序列机的设计思路，可以整理得到每个信号发生变化时对应的时刻以及此时对应的计数器的值。表 25.4 为依照线性序列机的设计思想，整理得到的每个信号发生变化时对应的时刻以及此时对应的计数器的值。其中 CS_N 为芯片状态标志信号，SCLK 为芯片时钟输入脚，DIN 为芯片串行数据输入，DOUT 为芯片串行数据输出。

表 25.4 时间点对应信号操作表

计数器值	对应信号操作
0	CS_N = 0;
1	SCLK = 0; DIN=0;
2	SCLK = 1;
3	SCLK = 0;
4	SCLK = 1;
5	SCLK = 0; DIN=ADD2;
6	SCLK = 1;
7	SCLK = 0; DIN=ADD1
8	SCLK = 1;
9	SCLK = 0; DIN=ADD0;
10	SCLK = 1;r_data[11]=DOUT;
11	SCLK = 0;
12	SCLK = 1; r_data[10]=DOUT;
13	SCLK = 0;
14	SCLK = 1; r_data[9]=DOUT;
15	SCLK = 0;
16	SCLK = 1; r_data[8]=DOUT;
17	SCLK = 0;

18	SCLK = 1; r_data[7]=DOUT;
19	SCLK = 0;
20	SCLK = 1; r_data[6]=DOUT;
21	SCLK = 0;
22	SCLK = 1; r_data[5]=DOUT;
23	SCLK = 0;
24	SCLK = 1; r_data[4]=DOUT;
25	SCLK = 0;
26	SCLK = 1; r_data[3]=DOUT;
27	SCLK = 0;
28	SCLK = 1; r_data[2]=DOUT;
29	SCLK = 0;
30	SCLK = 1; r_data[1]=DOUT;
31	SCLK = 0;
32	SCLK = 1; r_data[0]=DOUT;
33	CS_N = 1;

线性序列机计数器的控制逻辑判断依据，如表 4.17.6 所示。

基本条件	对 SCLK_GEN_CNT 操作
SCLK_GEN_CNT < 33	SCLK_GEN_CNT<= SCLK_GEN_CNT+ 1'b1;
SCLK_GEN_CNT = 33	SCLK_GEN_CNT<= 0;

表 4.17.6 计数器功能判断条件

以上就是通过线性序列机设计接口时序的一个典型案例，可以看到，线性序列机可以大大简化设计思路。线性序列机的设计思想就是使用一个计数器不断计数，由于每个计数值都会对应一个时间，那么当该时间符合需要操作信号的时刻时，就对该信号进行操作。这样，就能够轻松的设计出各种时序接口了。

5.3.4 基于线性序列机的 ADC 驱动设计

ADC128S022 接口逻辑模块图如图 25.5 所示。

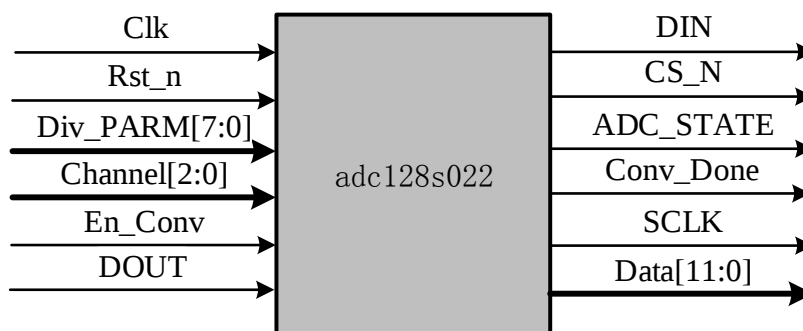


图 25.5 ADC 模块接口示意图

每个端口功能描述如表 25.5 所示。

表 25.5 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
Div_PARM[7:0]	I	时钟分频设置，实际 SCLK 时钟频率 = Clk / (DIV_PARAM * 2)
Channel[2:0]	I	通道选择
En_Conv	I	使能单次转换，该信号为单周期有效，高脉冲使能一次转换
DOUT	I	ADC 转换结果，由 ADC 输给 FPGA
DIN	O	ADC 控制信号，由 FPGA 发送通道控制字给 ADC
CS_N	O	ADC 串行数据接口使能信号
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
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
SCLK	O	ADC 串行数据接口时钟信号
Data[11:0]	O	ADC 转换结果

在每次使能转换的时候，寄存当前状态通道选择 Channel 的值，防止在转换过程中该值发生变化对转换过程产生影响。

```
reg [2:0] r_Channel; //通道选择内部寄存器
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。

```
reg en; //转换使能信号
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;
```

在数据手册中 SCLK 的频率范围为 0.8~3.2MHz。这里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。根据表 25.4 可以看出，需要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。

```
reg [7:0]DIV_CNT;//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)) //时钟分频设置
        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;
```

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'd33)
        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。

```
reg [11:0]r_data; //转换结果读取内部寄存器
reg [5:0]SCLK_GEN_CNT;//SCLK2X 计数器
always@(posedge Clk or negedge Rst_n)
if(!Rst_n)begin
    SCLK <= 1'b1;CS_N <= 1'b1;DIN <= 1'b1;
```

```

end
else if(en) begin
    if(SCLK2X)begin
        case(SCLK_GEN_CNT)
            //本部分见下
        endcase
    end
    else ;
end
else
    CS_N <= 1'b1;

```

对照表 25.4 可得以下。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 12 次。

```

case(SCLK_GEN_CNT)
    6'd0:begin CS_N <= 1'b0; end
    6'd1:begin SCLK <= 1'b0; DIN <= 1'b0; end
    6'd2:begin SCLK <= 1'b1; end
    6'd3:begin SCLK <= 1'b0; end
    6'd4:begin SCLK <= 1'b1; end
    6'd5:begin SCLK <= 1'b0; DIN <= r_Channel[2];end
    6'd6:begin SCLK <= 1'b1; end
    6'd7:begin SCLK <= 1'b0; DIN <= r_Channel[1];end
    6'd8:begin SCLK <= 1'b1; end
    6'd9:begin SCLK <= 1'b0; DIN <= r_Channel[0];end
    6'd10,6'd12,6'd14,6'd16,6'd18,6'd20,6'd22,6'd24,6'd26,6'd28,6'd
        30,6'd32:
        begin
            SCLK <= 1'b1; r_data <= {r_data[10:0], DOUT}; end
    6'd11,6'd13,6'd15,6'd17,6'd19,6'd21,6'd23,6'd25,6'd27,6'd29,6'd
        31: begin SCLK <= 1'b0; end
    6'd33:begin CS_N <= 1'b1; end//将转换结果输出
    default:begin CS_N <= 1'b1; end
endcase

```

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。

```

always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)begin
        Data <= 12'd0;
        Conv_Done <= 1'b0;
    end else if(en && SCLK2X && (SCLK_GEN_CNT == 6'd33))begin
        Data <= r_data;
    end

```

```

        Conv_Done <= 1'b1;
    end else begin
        Data <= Data;
        Conv_Done <= 1'b0;
    end

```

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。

```

assign ADC_State = CS_N;

```

5.3.5 仿真及板级测试

为了测试模块功能，模拟 ADC 芯片的输出。这里用 Sin3e 产生一个正弦波文件，位宽 12 位，个数为 4096，并以 sin_12bit.txt 保存当前工程下的 simulation 目录下的 modelsim 文件夹。

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>",<存储器名>);

\$readmemb("<数据文件名>",<存储器名>,<起始地址>);

\$readmemb("<数据文件名>",<存储器名>,<起始地址>,<结束地址>);

首先定义文件存放位置。

```

`define sin_data_file "./sin_12bit.txt"

```

定义 4096 个 12 位的存储单元，然后将波形数据读取到存储器中。

```

reg[11:0] memory[4095:0]; //测试波形数据存储空间
initial $readmemb(`sin_data_file,memory);

```

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。

```

integer i;
initial begin
    Rst_n = 0; Channel = 0; En_Conv = 0;
    DOUT = 0; address = 0; #101;
    Rst_n = 1; #100;
    Channel = 5;
    for(i=0;i<3;i=i+1)begin
        for(address=0;address<4095;address=address+1)begin
            En_Conv = 1;
            #20;
            En_Conv = 0;
            gene_DOUT(memory[address]);
            @(posedge Conv_Done); //等待转换完成信号
            #200;
        end
    end
end

```

```
        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 [15:0] vdata;
    reg [4:0] cnt;
    begin
        cnt = 0;
        wait(!CS_N);
        while(cnt<16)begin
            @(negedge SCLK) DOUT = vdata[15-cnt];
            cnt = cnt + 1'b1;
        end
    end
endtask
```

全编译后进行仿真，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 33d 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。

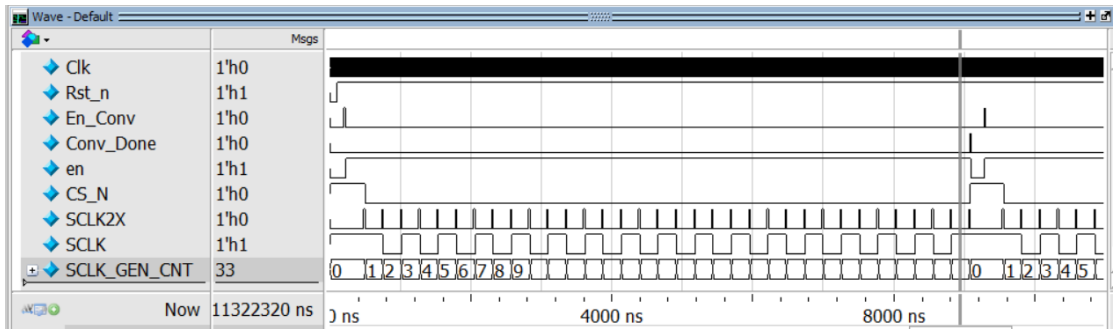


图 25.6 功能仿真波形

DIN 为 FPGA 控制 ADC 通道选择的信号，这里选择的通道五也就是 ADD[2:0]=101b，可以看出图 25.7 每个测量循环中发送 DIN 状态均一致。

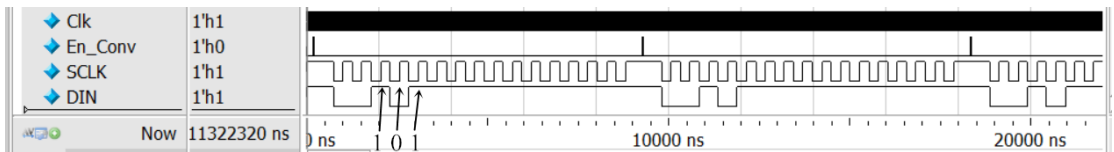


图 25.7 通道选择信号仿真波形

查看 DOUT 串行数据输出信号，在每个 SCLK 上升沿 DOUT 均会输出一位数据。在第

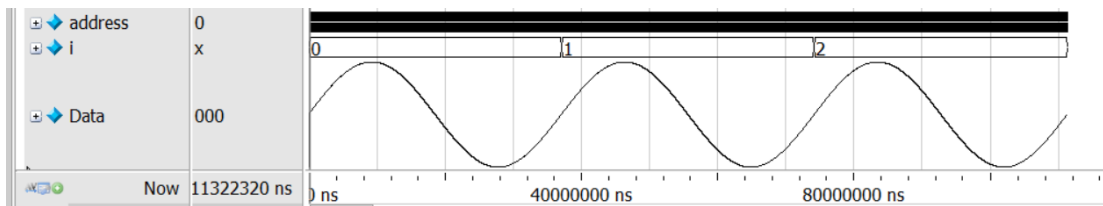


图 25.11 采样数据波形

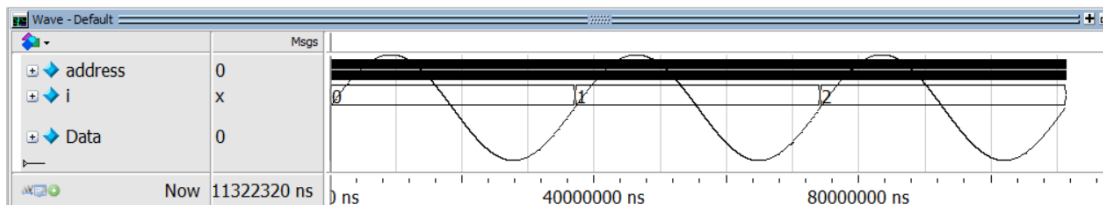


图 25.12 采样数据偏移波形

为了进行板级仿真，新建测试 ADC_test 顶层文件，在顶层文件中例化 ADC 以及 DAC 驱动，并设置好相关使能以及标志信号。再分别生成两个 ISSP 文件，其中驱动 DAC 电压数据的命名为 ISSP_DAC（源位宽为 16），采集 ADC 电压以及通道设置的为 ISSP_ADC（源位宽为 3，探针位宽为 12）。分配引脚并全编译后，下载程序，并启动 ISSP。

1. DAC 的模拟电压 A 通道输出端(AC620 开发板上标注丝印为 DA)连接 ADC 的 A0 输入端。如图 25.13 所示，使得 DAC 输出电压为 0，可测得电压为 0。

Index	Type	Alias	Name	Data	-16	-14	-12	-10	-8	-6	-4
S[15..0]			source[15..0]	C000h							
P[11..0]			probe[11..0]	000h	000h	006h	008h	000h	008h	003h	000h
S[2..0]			source[2..0]	0							0

图 25.13 联合测试图一

2. DAC 的模拟电压 B 通道(AC620 开发板上标注丝印为 DB)连接 ADC 的 A0 输入端。此时更新 DAC 的模拟电压 B 通道的输出值，如图 25.14，可以看出 A0 测量数据保持 0 不变。

Index	Type	Alias	Name	Data	-8	-7	-6	-5	-4	-3	-2	-1
S[15..0]			source[15..0]	47FFh								
P[11..0]			probe[11..0]	001h	003h		008h		00Ch		008h	
S[2..0]			source[2..0]	0								0

图 25.14 联合测试图二

3. DAC 的 DB 输出端接 ADC 的 A1 输入端。此时更新 DB 值，其理论输出电压为 $7FF/FFF \times 4.096 = 2.048V$ ，如图 25.15 所示，使能 A1 测量端可测得输入电压为 $2535/4096 \times 3.3 = 2.04V$ ，在误差允许范围内。

Index	Type	Alias	Name	Data	-8	-7	-6	-5	-4	-3	-2	-1
S[15..0]			source[15..0]	47FFh								
P[11..0]			probe[11..0]	2529	2528		2535		2525		2529	
S[2..0]			source[2..0]	1								1

图 25.15 联合测试图三

4. DAC 的 DB 输出端接 ADC 的 A1 输入端。此时更新 DB 值，其理论输出电压为 $BD3/FFF \times 4.096 = 3.072V$ ，如图 25.16 所示，使能 A1 测量端可测得输入电压为 $3815/4096 \times 3.3 = 2.07V$ ，在误差允许范围内。

0					-8	-7	-6	-5	-4	-3	-2	-1
Index	Type	Alias	Name	Data								
S[15..0]			source[15..0]	4BD3h								
1					-8	-7	-6	-5	-4	-3	-2	-1
Index	Type	Alias	Name	Data								
P[11..0]			probe[11..0]	3815	3815	3811	3815	3804	3808	3815	3811	
S[2..0]			source[2..0]	1								1

图 25.16 联合测试图四

同理可测试的其他 DAC 输出电压以及 ADC 输入测量电压值。此处换算时需注意，DAC 电路输出电压范围为 0~4V，ADC 电路测量电压范围为 0~3.3V。

本章完成了 ADC 驱动的设计，并进行了功能仿真以及与 DAC 芯片的联合板级调试。可结合第三章中的数码管驱动实现一个 8 通道的数字电压表，具体实现过程可参见第六章相关章节。