双路高速 DA 实验

DAC（Digital to Analog Converter，即数模转换器）是大多数系统中必不可少的组成部件，用于将离散的数字信号转换成连续的模拟信号，它们是连接模电电路和数字电路必不可少的桥梁。在很多场合下，DAC 的转换速度甚至直接决定了整个系统的运行速度。本章我们将使用高速 DA 芯片实现数模转换，产生正弦波模拟电压信号。

本章包括以下几个部分：

1.1 简介

1.2 实验任务

1.3 硬件设计

1.4 程序设计

1.5 下载验证

1.1 简介

本章我们使用的双路 DA 模块是正点原子推出的一款双路高速数模转换模块（ATK_DUAL_HS_DA），高速 DA 转换芯片是由思瑞浦公司生产的 3PD5651E 芯片。

ATK_HS_AD_DA 模块的硬件结构图如下图所示。

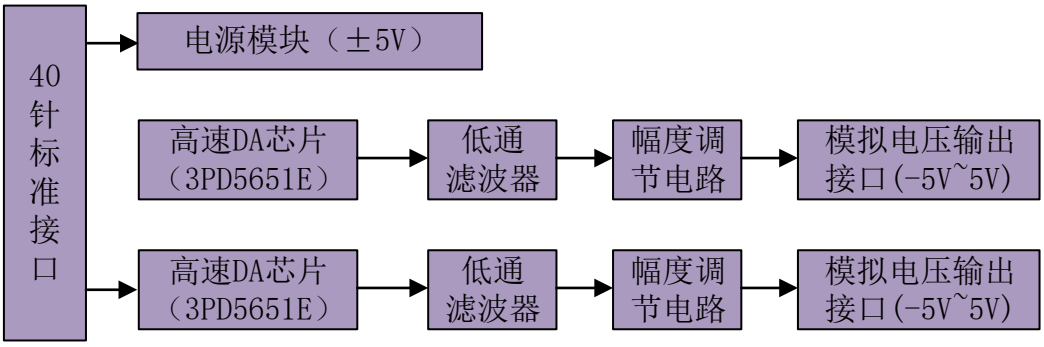

图 1.1.1 ATK_DUAL_HS_DA 模块硬件结构图

由上可知，3PD5651E 芯片输出的是一对差分电流信号，为了防止受到噪声干扰，电路中接入了低通滤波器，然后通过高性能和高带宽的运放电路，实现差分变单端以及幅度调节等功能，使整个电路性能得到了最大限度的提升，最终输出的模拟电压范围是-5V~+5V。

下面来介绍下这款芯片。

3PD5651E 是 3PEAK 公司（思瑞浦微电子科技股份有限公司）生产的 DAC 系列数模转换器，具有高性能、低功耗的特点。3PD5651E 的数模转换位数为 10 位，最大转换速度为 125MSPS（每秒采样百万次，Million Samples per Second）。

3PD5651E 的内部功能框图如下图所示：

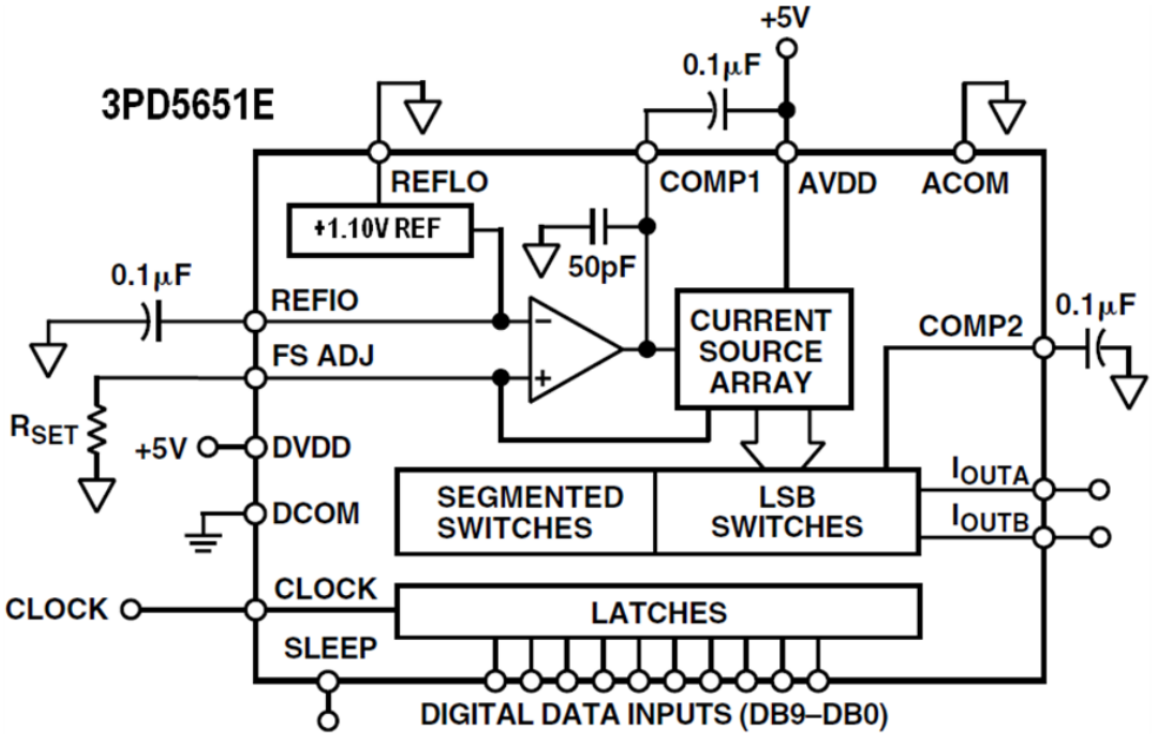

Figure 2. Functional Block Diagram

图 1.1.2 内部功能框图

3PD5651E 在时钟（CLOCK）的驱动下工作，内部集成了+1.1V 参考电压(+1.10V REF)、运算放大器、

电流源（CURRENT SOURCE ARRAY）和锁存器（LATCHES）。两个电流输出端 IOUTA 和 IOUTB 为一对差分电流，当输入数据为 0（DB9~DB0=10’h000）时，IOUTA 的输出电流为 0，而 IOUTB 的输出电流达到最大，最大值的大小跟参考电压有关；当输入数据全为高点平（DB9~DB0=10’h3ff）时，IOUTA 的输出电流达到最大，最大值的大小跟参考电压有关，而 IOUTB 的输出电流为 0。

3PD5651E 必须在时钟的驱动下才能把数据写入片内的锁存器中，其触发方式为上升沿触发，3PD5651E 的时序图如下图所示：

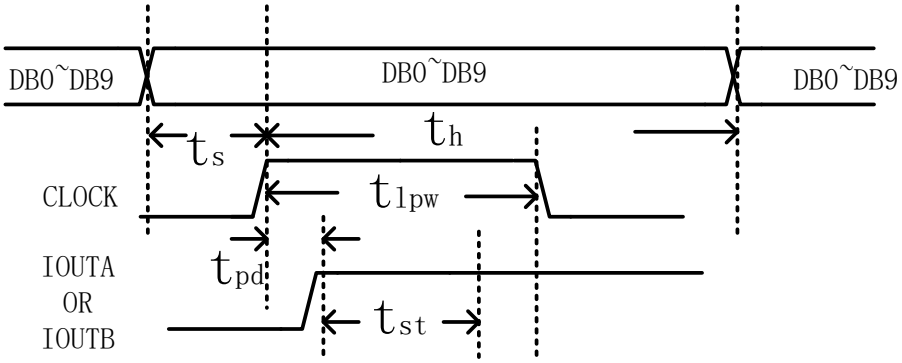

图 1.1.3 芯片时序图

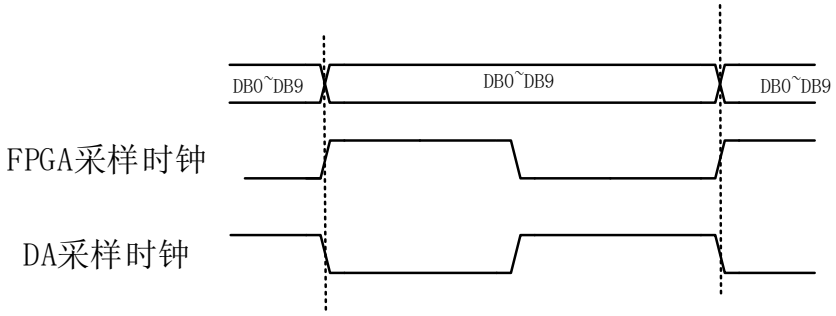

图 1.1.4 FPGA 内部时序

如图 1.1.3 中的 DBO-DB9 和 CLOCK 是 3PD5651E 的 10 位输入数据和为输入时钟，IOUTA 和 IOUTB 为 3PD5651E 输出的电流信号。由图 1.1.3 可知，数据在时钟的上升沿锁存，因此我们可以在时钟的下降沿发送数据，这样使 DA 芯片在数据的中央采样，保证数据采样的准确性，如图 1.1.4 所示。需要注意的是，CLOCK 的时钟频率越快，3PD5651E 的数模转换速度越快，3PD5651E 的时钟频率最快为 125Mhz。

IOUTA 和 IOUTB 为 3PD5651E 输出的一对差分电流信号，通过外部电路低通滤波器与运放电路输出模拟电压信号，电压范围是-5V 至+5V 之间。当输入数据等于 0 时，3PD5651E 输出的电压值为 5V；当输入数据等于 10’h3ff 时，3PD5651E 输出的电压值为-5V。

3PD5651E 是一款数字信号转模拟信号的器件，内部没有集成 DDS（Direct Digital Synthesizer，直接数字式频率合成器）的功能，但是可以通过控制 3PD5651E 的输入数据，使其模拟 DDS 的功能。例如，我们使用 3PD5651E 输出一个正弦波模拟电压信号，那么我们只需要将 3PD5651E 的输入数据按照正弦波的波形变化即可，下图为 3PD5651E 的输入数据和输出电压值按照正弦波变化的波形图。

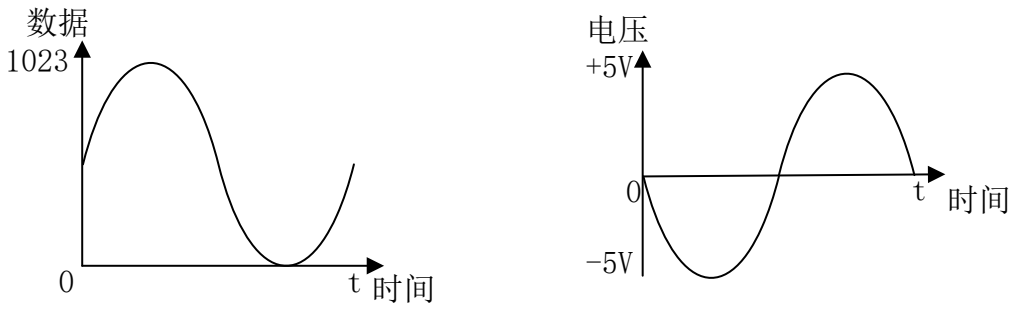

图 1.1.5 3PD5651E 正弦波数据（左）、电压值（右）

由上图可知，数据在 0 至 1023 之间按照正弦波的波形变化，最终得到的电压也会按照正弦波波形变化，当输入数据重复按照正弦波的波形数据变化时，那么 3PD5651E 就可以持续不断的输出正弦波的模拟电压波形。需要注意的是，最终得到的 3PD5651E 的输出电压变化范围由其外部电路决定的，当输入数据为 0 时，3PD5651E 输出+5V 的电压；当输入数据为 1023 时，3PD5651E 输出-5V 的电压。

由此可以看出，只要输入的数据控制的得当，AD9708 可以输出任意波形的模拟电压信号，包括正弦波、方波、锯齿波、三角波等波形。

1.2 实验任务

本节实验任务是使用达芬奇开发板及双路高速 DA 扩展模块（ATK_DUAL_HS_DA 模块）实现数模转换。首先利用 FPGA 产生正弦波变化的数字信号，经过 DA 芯片后转换成模拟信号，然后通过示波器观察模拟信号的波形是否按照正弦波波形变化。

1.3 硬件设计

ATK_DUAL_HS_DA 模块由 2 个型号为 3PD5651E 的 DA 转换芯片组成。3PD5651E 的原理图如下图所示。

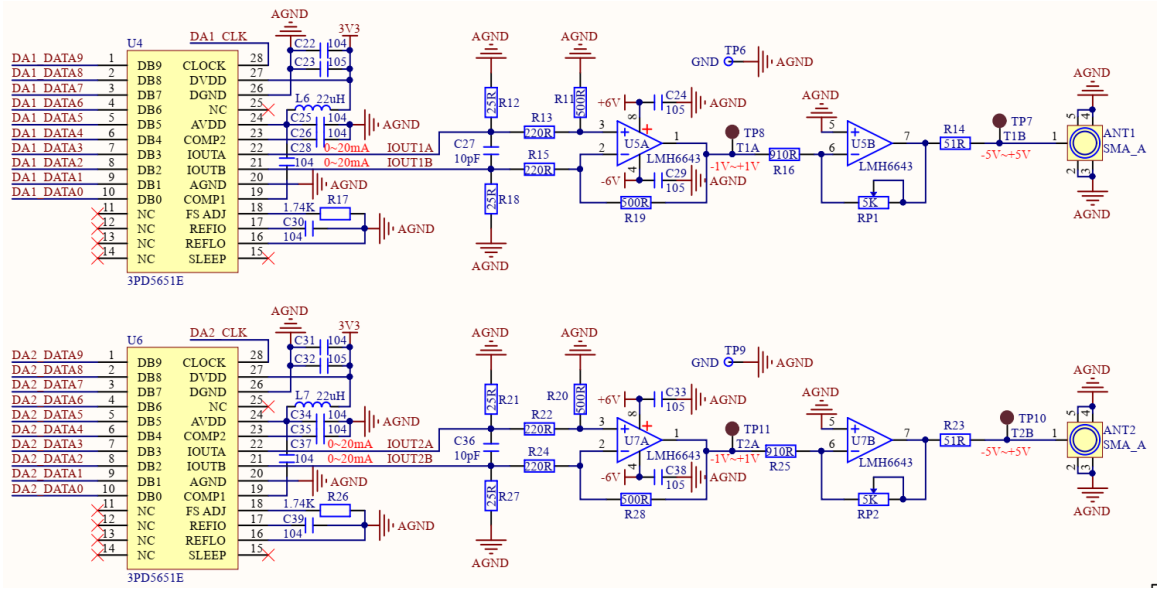

图 1.3.1 芯片原理图

由上图可知，3PD5651E 输出的一对差分电流信号先经过滤波器，再经过运放电路得到一个单端的模拟电压信号。图中右侧的 RP1 为滑动变阻器，可以调节输出的电压范围，推荐通过调节滑动变阻器，使输出的电压范围在-5V 至+5V 之间，从而达到 DA 转换芯片的最大转换范围。

ATK_DUAL_HS_DA 模块的实物图如下图所示。

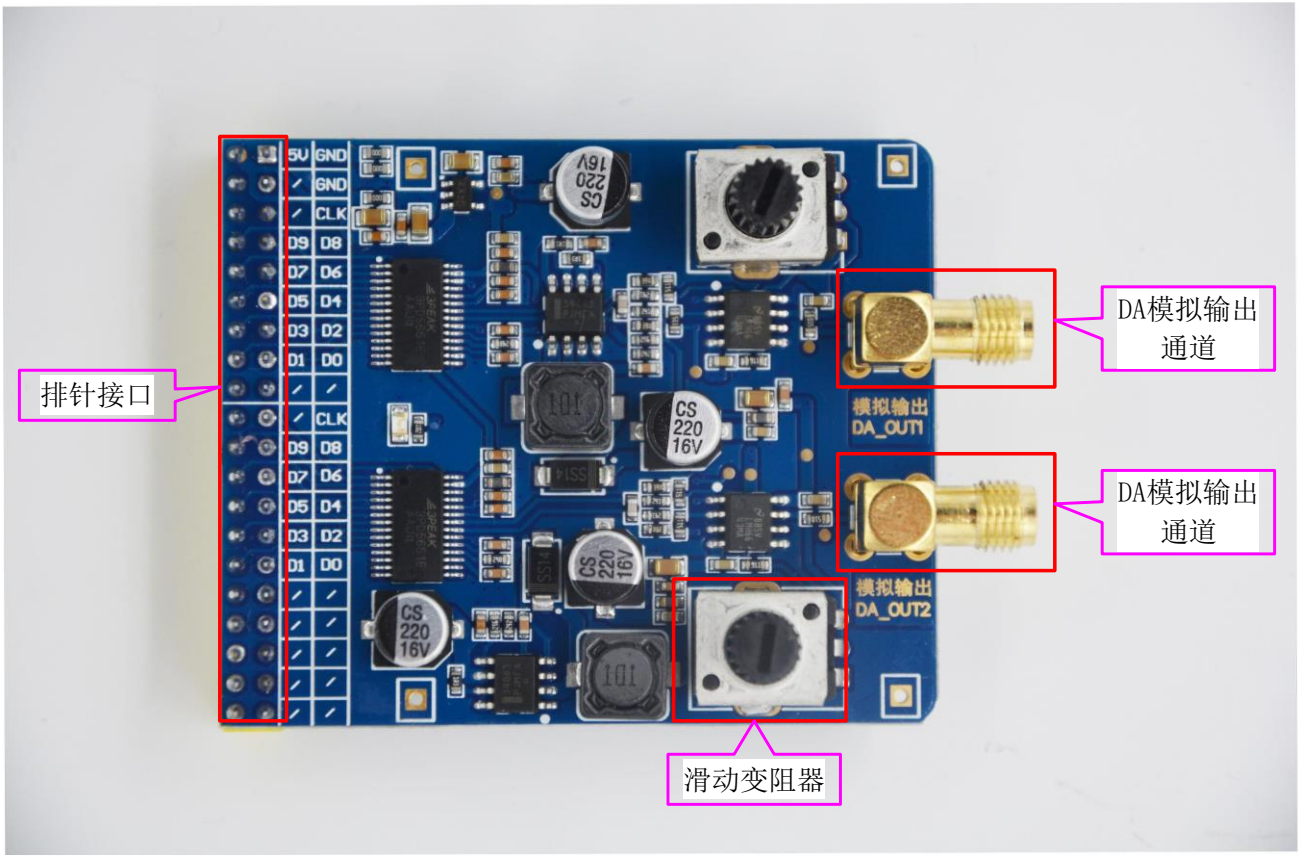

图 1.3.2 ATK_DUAL_HS_DA 模块实物图

本实验中，各端口信号的管脚分配如下表所示。

表格 1.3.1 双路高速DA转换实验管脚分配

信号名	方向	管脚	端口说明	电平标准
sys_clk	input	R4	系统时钟，50Mhz	LVCMOS33
sys_rst_n	input	U2	系统复位，低有效	LVCMOS33
da_clk	output	Y7	DA（3PD5651E）驱动时钟	LVCMOS33
da_data[0]	output	Y8	输出给DA的数据	LVCMOS33
da_data[1]	output	AA8	输出给DA的数据	LVCMOS33
da_data[2]	output	AB8	输出给DA的数据	LVCMOS33
da_data[3]	output	AB6	输出给DA的数据	LVCMOS33
da_data[4]	output	V7	输出给DA的数据	LVCMOS33
da_data[5]	output	AB7	输出给DA的数据	LVCMOS33
da_data[6]	output	U7	输出给DA的数据	LVCMOS33
da_data[7]	output	V8	输出给DA的数据	LVCMOS33

da_data[8]	output	Y9	输出给DA的数据	LVCMOS33
da_data[9]	output	W9	输出给DA的数据	LVCMOS33
da_clk1	output	R14	DA（3PD5651E）驱动时钟	LVCMOS33
da_data1[0]	output	P15	输出给DA的数据	LVCMOS33
da_data1[1]	output	R16	输出给DA的数据	LVCMOS33
da_data1[2]	output	P17	输出给DA的数据	LVCMOS33
da_data1[3]	output	N17	输出给DA的数据	LVCMOS33
da_data1[4]	output	R18	输出给DA的数据	LVCMOS33
da_data1[5]	output	V20	输出给DA的数据	LVCMOS33
da_data1[6]	output	T18	输出给DA的数据	LVCMOS33
da_data1[7]	output	V19	输出给DA的数据	LVCMOS33
da_data1[8]	output	U17	输出给DA的数据	LVCMOS33
da_data1[9]	output	U18	输出给DA的数据	LVCMOS33

对应的 XDC 约束语句如下所示：

```
create_clock -period 20.000 -name sys_clk [get_ports sys_clk]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets sys_clk]
set_property PACKAGE_PIN R4 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports sys_rst_n]
set_property PACKAGE_PIN U2 [get_ports sys_rst_n]

set_property PACKAGE_PIN Y8 [get_ports {da_data[0]}]
set_property PACKAGE_PIN AA8 [get_ports {da_data[1]}]
set_property PACKAGE_PIN AB8 [get_ports {da_data[2]}]
set_property PACKAGE_PIN AB6 [get_ports {da_data[3]}]
set_property PACKAGE_PIN V7 [get_ports {da_data[4]}]
set_property PACKAGE_PIN AB7 [get_ports {da_data[5]}]
set_property PACKAGE_PIN U7 [get_ports {da_data[6]}]
set_property PACKAGE_PIN V8 [get_ports {da_data[7]}]
set_property PACKAGE_PIN Y9 [get_ports {da_data[8]}]
set_property PACKAGE_PIN W9 [get_ports {da_data[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[4]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports da_clk]
set_property PACKAGE_PIN Y7 [get_ports da_clk]

set_property PACKAGE_PIN P15 [get_ports {da_data1[0]}]
set_property PACKAGE_PIN R16 [get_ports {da_data1[1]}]
set_property PACKAGE_PIN P17 [get_ports {da_data1[2]}]
set_property PACKAGE_PIN N17 [get_ports {da_data1[3]}]
set_property PACKAGE_PIN R18 [get_ports {da_data1[4]}]
set_property PACKAGE_PIN V20 [get_ports {da_data1[5]}]
set_property PACKAGE_PIN T18 [get_ports {da_data1[6]}]
set_property PACKAGE_PIN V19 [get_ports {da_data1[7]}]
set_property PACKAGE_PIN U17 [get_ports {da_data1[8]}]
set_property PACKAGE_PIN U18 [get_ports {da_data1[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {da_data1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports da_clk1]
set_property PACKAGE_PIN R14 [get_ports da_clk1]
```

1.4 程序设计

根据本章的实验任务，FPGA 需要连续输出正弦波波形的数据，才能使 3PD5651E 连续输出正弦波波形的模拟电压，如果通过编写代码使用三角函数公式运算的方式输出正弦波数据，那么程序设计会变得非常复杂。在工程应用中，一般将正弦波波形数据存储在 RAM 或者 ROM 中，由于本次实验并不需要写数据到 RAM 中，因此我们将正弦波波形数据存储在只读的 ROM 中，直接读取 ROM 中的数据发送给 DA 转换芯片即可。

图 1.4.1 是根据本章实验任务画出的系统框图。ROM 里面事先存储好了正弦波波形的数据，DA 数据发送模块从 ROM 中读取数据，将数据和时钟送到 3PD5651E 芯片的输入数据端口和输入时钟端口。

双路高速 DA 实验的系统框图如图 1.4.1 所示：

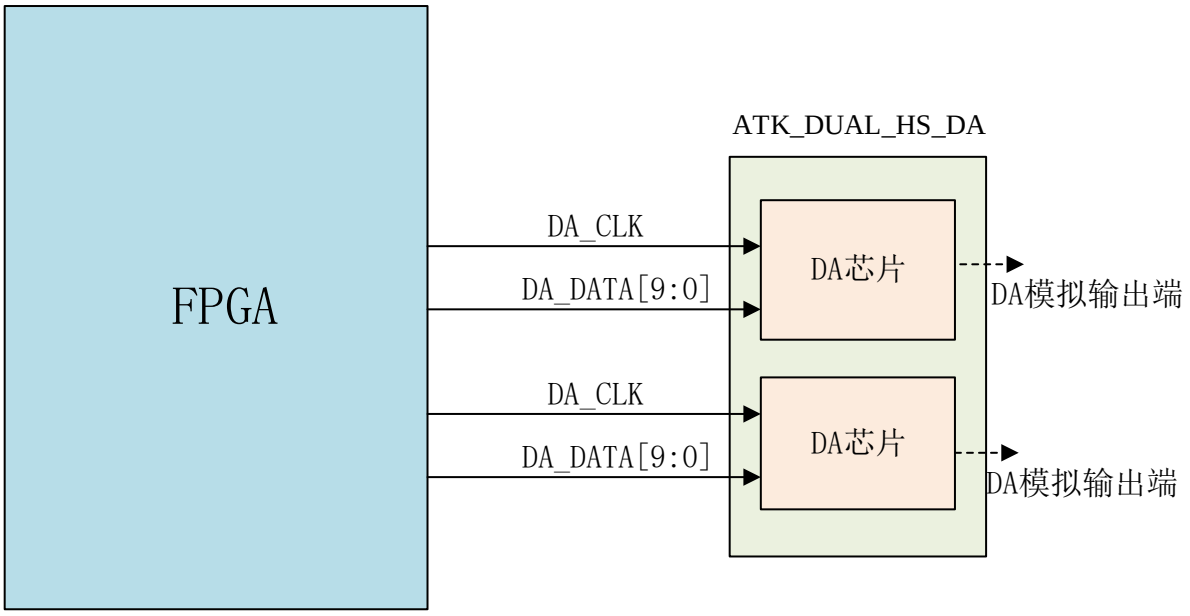

图 1.4.1 双路高速 DA 系统框图

顶层模块的原理图如下图所示：

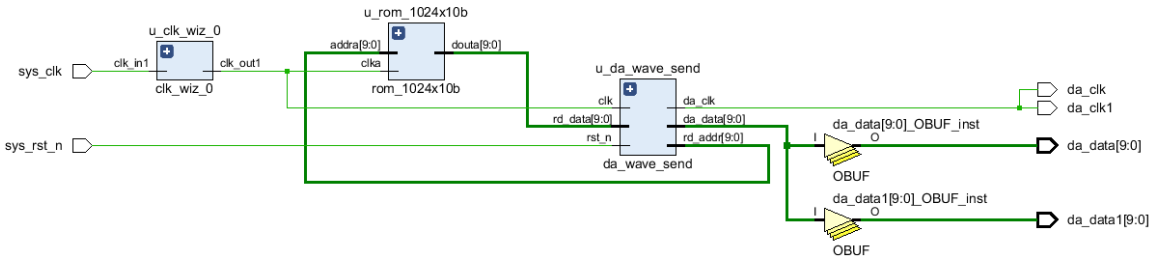

图 1.4.2 顶层模块原理图

FPGA 顶层模块（hs_dual_da）例化了以下三个模块：DA 数据发送模块（da_wave_send）、ROM 波形存储模块（rom_1024x10b）和时钟模块（clk_wiz_0）。

DA 数据发送模块（da_wave_send）：DA 数据发送模块输出读 ROM 地址，将输入的 ROM 数据发送至 DA 转换芯片的数据端口。

ROM 波形存储模块(rom_1024x10b)：ROM 波形存储模块由 Vivado 软件自带的 Block Memory Generator IP 核实现，其存储的波形数据可以使用波形转存储文件的上位机来生成.coe 文件。

顶层模块的代码如下：

```
1  module hs_dual_da(
2      input                sys_clk     ,  //系统时钟
3      input                sys_rst_n   ,  //系统复位，低电平有效
4      //DA 接口
5      output               da_clk      ,  //DA 采样时钟
6      output    [9:0]      da_data     ,  //DA 采样数据
7      output               da_clk1     ,  //DA 采样时钟
8      output    [9:0]      da_data1       //DA 采样数据
9  );
```

```
10
11 //wire define
12 wire      [9:0]    rd_addr;              //ROM 地址
13 wire      [9:0]    rd_data;              //ROM 数据
14
15 //*****************************************************
16 //**                    main code
17 //*****************************************************
18
19 assign  da_clk1 = da_clk;
20 assign  da_data1 = da_data;
21
22 //时钟模块
23 clk_wiz_0  u_clk_wiz_0(
24  .clk_in1 (sys_clk),
25  .clk_out1 (clk)
26 );
27
28 //DA 发送模块
29 da_wave_send u_da_wave_send(
30     .clk          (clk),
31     .rst_n        (sys_rst_n),
32     .rd_data      (rd_data),
33     .rd_addr      (rd_addr),
34     .da_clk       (da_clk),
35     .da_data      (da_data)
36     );
37
38 //ROM 模块
39 rom_1024x10b  u_rom_1024x10b(
40     .addra      (rd_addr),
41     .clka       (clk),
42     .douta      (rd_data)
43 );
44
45 endmodule
```

在代码的第 23 至 26 行例化了时钟模块，倍频出 125M 时钟给 DA 芯片采样用。

DA 数据发送模块输出的读 ROM 地址（rd_addr）连接至 ROM 模块的地址输入端，ROM 模块输出的数据（rd_data）连接至 DA 数据发送模块的数据输入端，从而完成了从 ROM 中读取数据的功能。

在代码的第 39 至 43 行例化了 ROM 模块，由 Block Memory Generator IP 核配置生成。

我们在前面说过，ROM 中存储的波形数据可以使用上位机波形转 COE 软件生成，在这里我们介绍一个简单易用的波形转 COE 工具的使用方法，该工具位于开发板所随附的资料“6_软件资料/1_软件

/WaveToMem”目录下，双击“WaveToMem_V1.2.exe”运行软件。

接下来我们对软件进行设置，如图 1.4.3 所示，这里对软件界面做个简单的介绍。

位宽：波形数据的位宽。由于 ATK_DUAL_HS_DA 模块的 DA 芯片数据位宽为 10 位，因此这里将位宽设成 10 位。

深度：一个波形周期包含了多少个数据量。这里将深度设置成 1024。需要说明的是，在用 Block Memory Generator IP 核生成 ROM 时，配置 ROM 的宽度和深度和上位机设置的位宽和深度保持一致。

波形频率设置：对波形倍频，倍数值越大，最终生成的波形频率越快（频率太高，可能导致波形失真），这里保持默认，即设置成 1 位。

波形类型：软件支持将正弦波、方波、锯齿波和三角波的波形转换成存储波形格式的文件。

生成文件：软件支持将波形转换成 COE（Vivado 软件支持的存储格式）和 MIF（Quartus 软件支持的存储格式）格式文件，这里保持默认，即选中 COE 文件格式。

然后点击“一键生成”按钮，在弹出的界面中选择 COE 文件的存放路径并输入文件名，这里将 COE 文件保存在工程的 sources_1\new 文件夹下。WaveToMem 转换过程中的软件界面如下图所示：

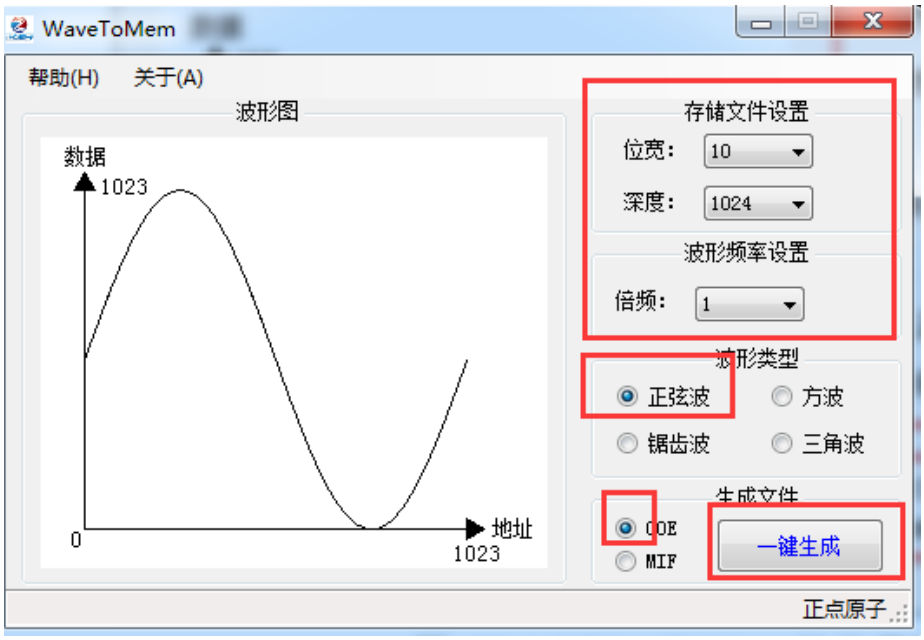

图 1.4.3 WaveToMem 软件界面

使用 Notepad++代码编辑器打开生成的 COE 文件后如下图所示：

```
1  MEMORY_INITIALIZATION_RADIX=10;
2  MEMORY_INITIALIZATION_VECTOR=
3    511
4    514
5    517
6    520
7    523
8    526
9    529
10   532
11   536
12   539
13   542
14   545
15   548
16   551
```

图 1.4.4 COE 文件打开界面

工程中创建了一个单端口 ROM，并命名为“rom_1024x10b”，在调用 Block Memory Generator IP 核时，“Basic”选项也配置如下图所示：

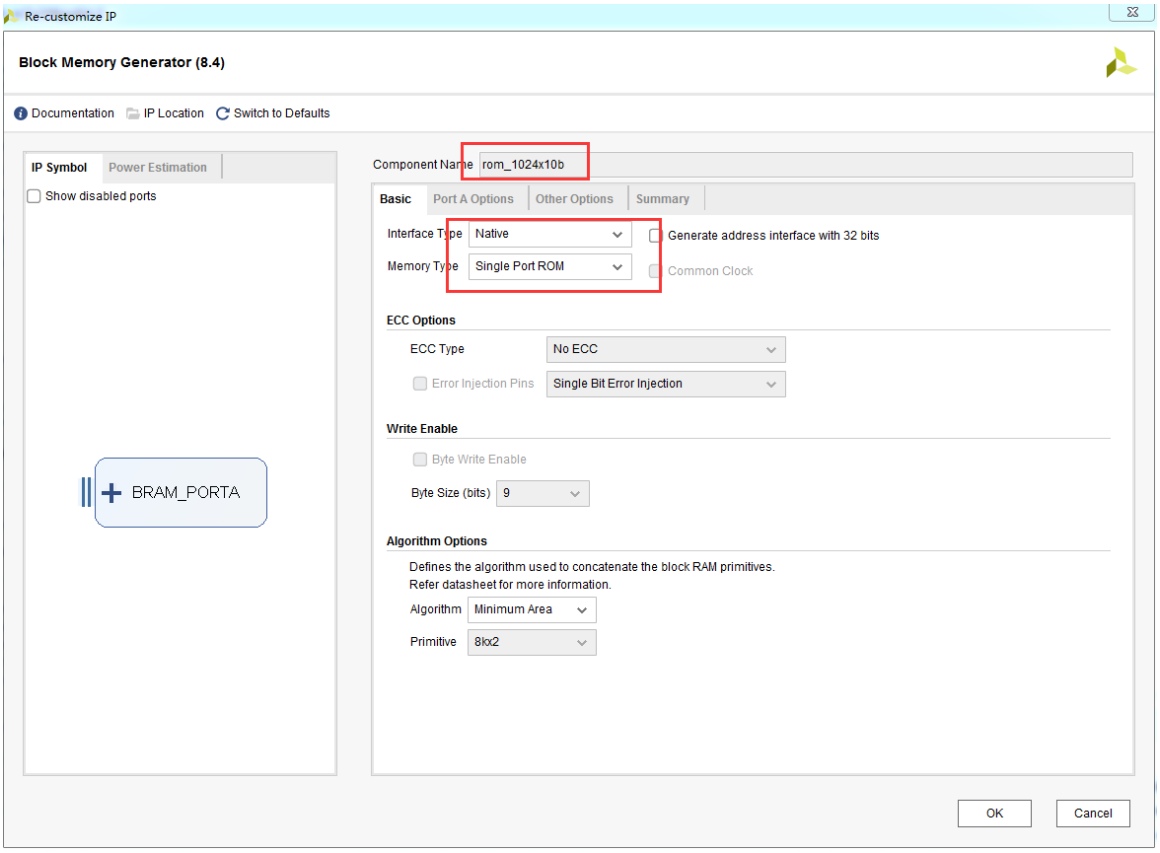

图 1.4.5 Block Memory Generator IP 核的 Basic 配置页面

我们将其接口类型设置为“Native”、Memory Type 设置为“Single Port ROM”，即单端口 ROM。“Port A Options”选项页的配置页面如下图所示：

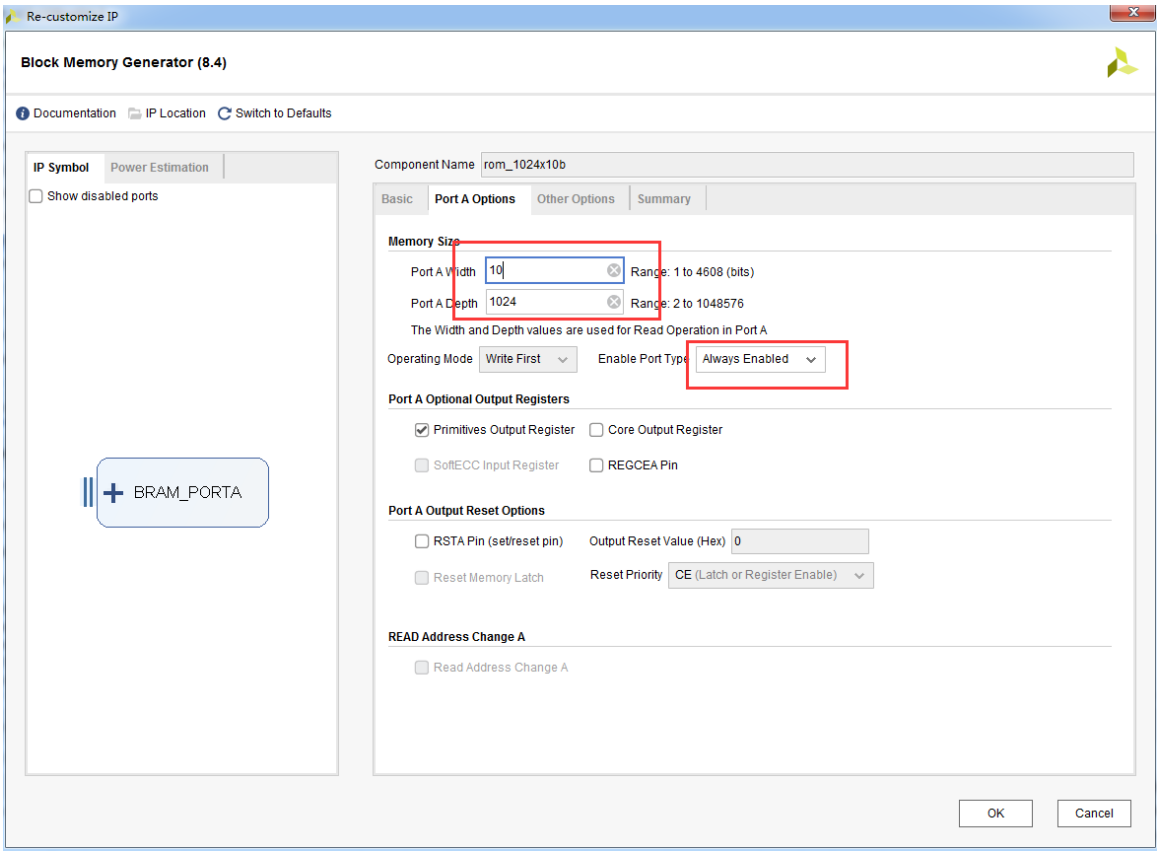

图 1.4.6 Block Memory Generator IP 核的 PortA Options 配置页面

我们将 PortA 的位宽设置为 10，深度设置为 1024，以存储上位机生成的 1024 个数据。此外，将使能引脚的类型设置为“Always Enabled”，即 ROM 一直处于使能的状态。

接下来配置“Other Options”选项页，加载刚才生成的.coe 文件，如下图所示：

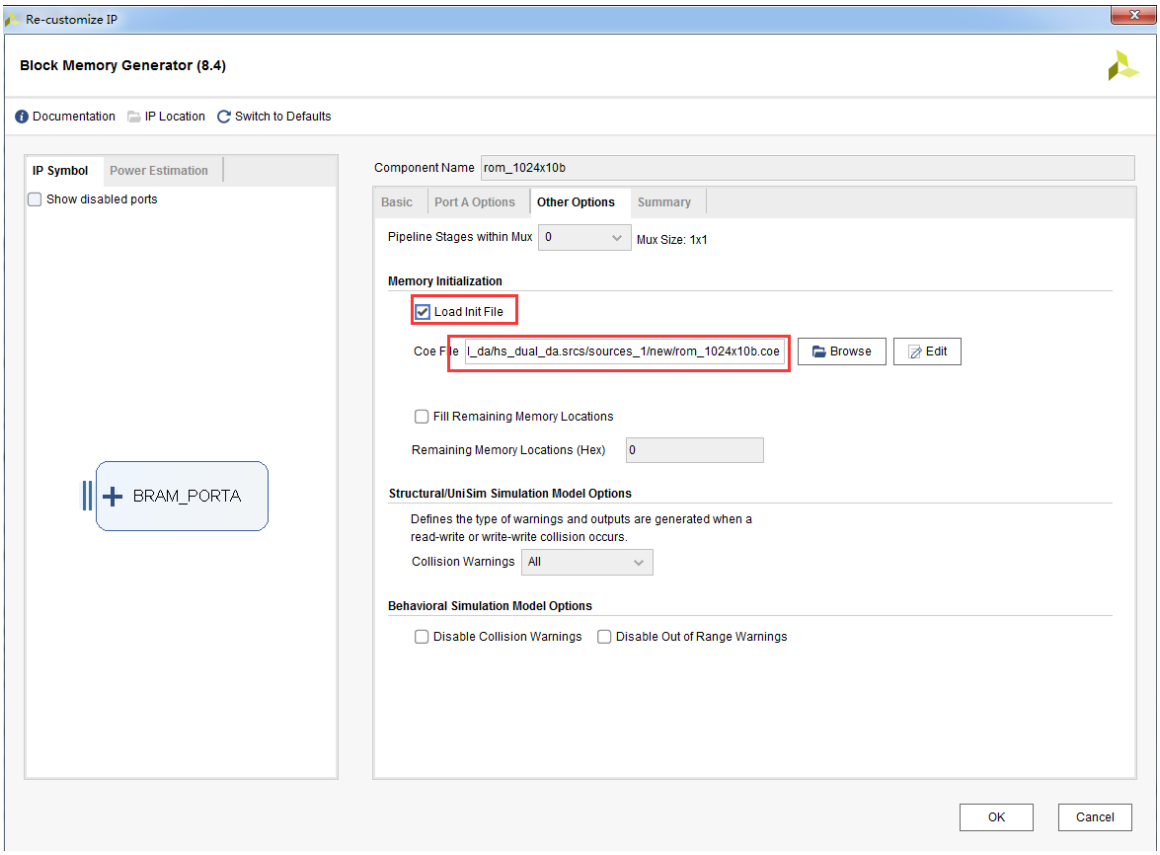

图 1.4.7 Block Memory Generator IP 核的 Other Options 配置页面

最后点击“OK”按钮完成 IP 核的配置。

DA 数据发送模块的代码如下：

```
module da_wave_send(
    input                clk    ,  //时钟
    input                rst_n  ,  //复位信号，低电平有效

    input        [9:0]   rd_data,  //ROM 读出的数据
    output  reg  [9:0]   rd_addr,  //读 ROM 地址
    //DA 芯片接口
    output               da_clk ,  //DA(AD9708)驱动时钟,最大支持 125Mhz 时钟
    output       [9:0]   da_data   //输出给 DA 的数据
    );

//parameter
//频率调节控制
parameter  FREQ_ADJ = 10'd5;  //频率调节,FREQ_ADJ 的越大,最终输出的频率越低,范围 0~255

//reg define
reg    [9:0]    freq_cnt  ;  //频率调节计数器

//*****************************************************
```

```
20 //**                     main code
21 //*************************************************
22
23 //数据 rd_data 是在 clk 的上升沿更新的，所以 DA 芯片在 clk 的下降沿锁存数据是稳定的时刻
24 //而 DA 实际上在 da_clk 的上升沿锁存数据,所以时钟取反,这样 clk 的下降沿相当于 da_clk 的上升沿
25 assign  da_clk = ~clk;
26 assign  da_data = rd_data;   //将读到的 ROM 数据赋值给 DA 数据端口
27
28 //频率调节计数器
29 always @(posedge clk or negedge rst_n) begin
30     if(rst_n == 1'b0)
31         freq_cnt <= 10'd0;
32     else if(freq_cnt == FREQ_ADJ)
33         freq_cnt <= 10'd0;
34     else
35         freq_cnt <= freq_cnt + 10'd1;
36 end
37
38 //读 ROM 地址
39 always @(posedge clk or negedge rst_n) begin
40     if(rst_n == 1'b0)
41         rd_addr <= 10'd0;
42     else begin
43         if(freq_cnt == FREQ_ADJ) begin
44             rd_addr <= rd_addr + 10'd1;
45         end
46     end
47 end
48
49 endmodule
```

在代码的第 14 行定义了一个参数 FREQ_ADJ（频率调节），可以通过控制频率调节参数的大小来控制最终输出正弦波的频率大小，频率调节参数的值越小，正弦波频率越大。频率调节参数调节正弦波频率的方法是通过控制读 ROM 的速度实现的，频率调节参数越小，freq_cnt 计数到频率调节参数值的时间越短，读 ROM 数据的速度越快，那么正弦波输出频率也就越高；反过来，频率调节参数越大，freq_cnt 计数到频率调节参数值的时间越长，读 ROM 数据的速度越慢，那么正弦波输出频率也就越低。由于 freq_cnt 计数器的位宽为 10 位，计数范围是 0~1023，所以频率调节参数 FREQ_ADJ 支持的调节范围是 0~1023，可通过修改 freq_cnt 计数器的位宽来修改 FREQ_ADJ 支持的调节范围。

WaveToMem 软件设置 ROM 深度为 1024，倍频系数为 1，而输入时钟为 125Mhz，那么一个完整的正弦波周期长度为 1024*8ns = 8192ns，当 FREQ_ADJ 的值为 0 时，即正弦波的最快输出频率为 1s/8192ns(1s = 1000000000ns) ≈ 122.0Khz。当我们把 FREQ_ADJ 的值设置为 5 时，一个完整的正弦波周期长度为 5120ns*(5+1) =49152ns，频率约为 20.35KHz。也可以在 WaveToMem 软件设置中增加倍频系数或者增加 AD 的驱动时钟来提高正弦波输出频率。

1.5 下载验证

将双路高速 DA 模块插入达芬奇开发板的 J3 扩展口（靠近数码管的位置），连接时注意扩展口电源引脚方向和开发板电源引脚方向一致，然后将下载器一端连接电脑，另一端与开发板上对应端口连接，最后连接电源线并打开电源开关。

达芬奇开发板硬件连接实物图如下图所示：

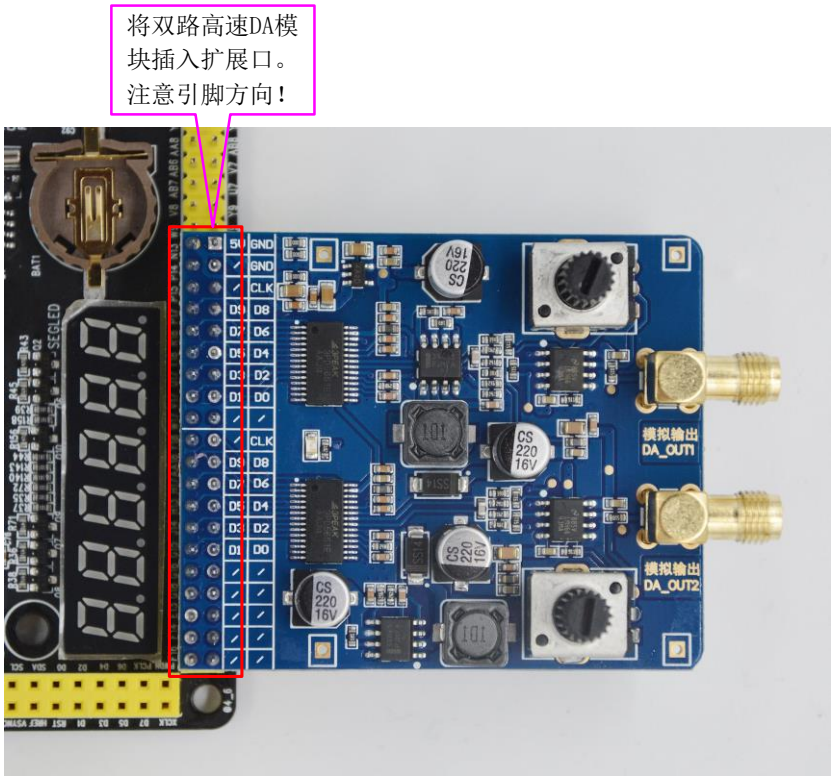

图 1.5.1 达芬奇开发板硬件连接实物图

将工程生成的比特流文件下载到达芬奇开发板中后，然后使用示波器测量 DA 输出通道的波形。首先将示波器带夹子的一端连接到开发板的 GND 位置（可使用杜邦线连接至开发板上的任一的 GND 管脚），然后将另一端探针插入高速 AD-DA 模块的 DA 通道中间的金属圆圈内（注意将红色的保护套拿掉），如图 1.5.2 所示。

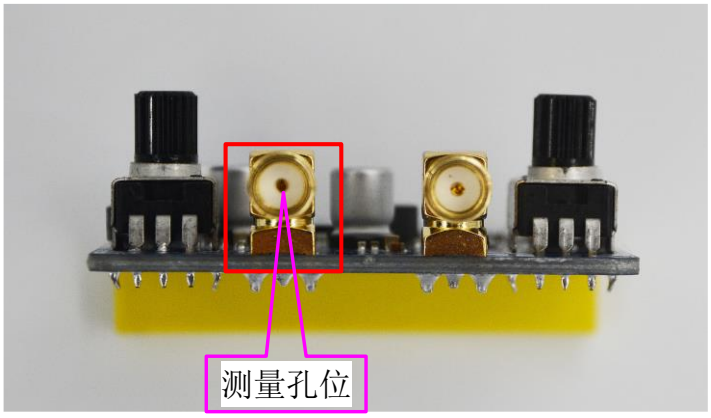

图 1.5.2 DA 模拟电压测量孔位

此时观察示波器可以看到正弦波的波形，如果观察不到波形，可查看示波器设置是否正确，可以尝试按下示波器的“AUTO”，再次观察示波器波形。示波器的显示界面如下图所示：

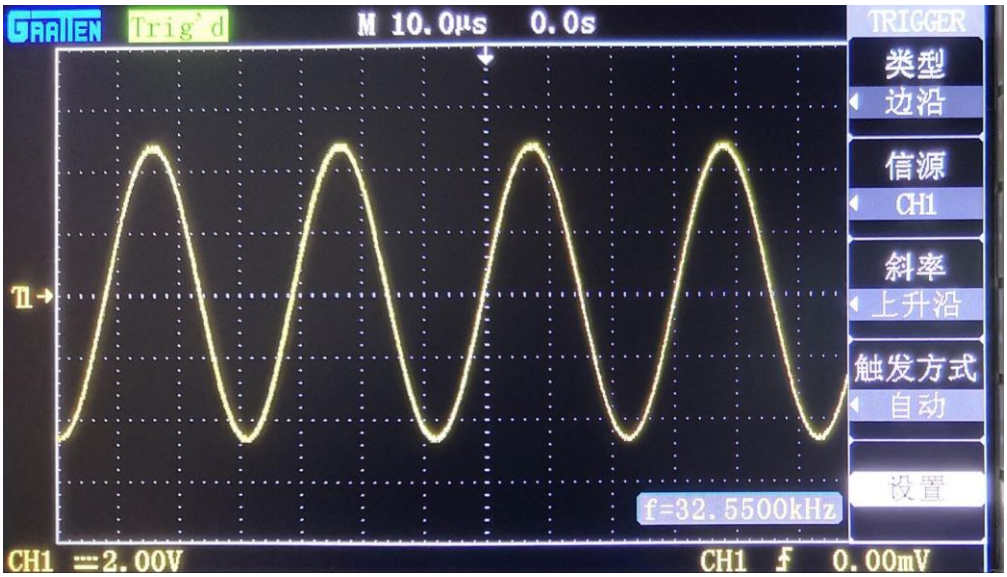

图 1.5.3 示波器显示界面