第26章　简易 DDS 信号发生器的设计与验证

26.1 章节导读

DDS 是直接数字式频率合成器（Direct Digital Synthesizer）的英文缩写，是一项关键的数字化技术。与传统的频率合成器相比，DDS 具有低成本、低功耗、高分辨率和快速转换时间等优点，广泛使用在电信与电子仪器领域，是实现设备全数字化的一个关键技术。作为设计人员，我们习惯称它为信号发生器，一般用它产生正弦、锯齿、方波等不同波形或不同频率的信号波形，在电子设计和测试中得到广泛应用。

本章节，我们将带领读者学习一下 DDS 的相关知识，运用所学知识，设计实现一个简易的 DDS 信号发生器，并上板验证。

26.2 理论学习

DDS 技术是一种全新的频率合成方法，其具有低成本、低功耗、高分辨率和快速转换时间等优点，对数字信号处理及其硬件实现有着很重要的作用。

DDS 的基本结构主要由相位累加器、相位调制器、波形数据表 ROM、D/A 转换器等四大结构组成，其中较多设计还会在数模转换器之后增加一个低通滤波器。DDS 结构示意图，具体见图 26-1。

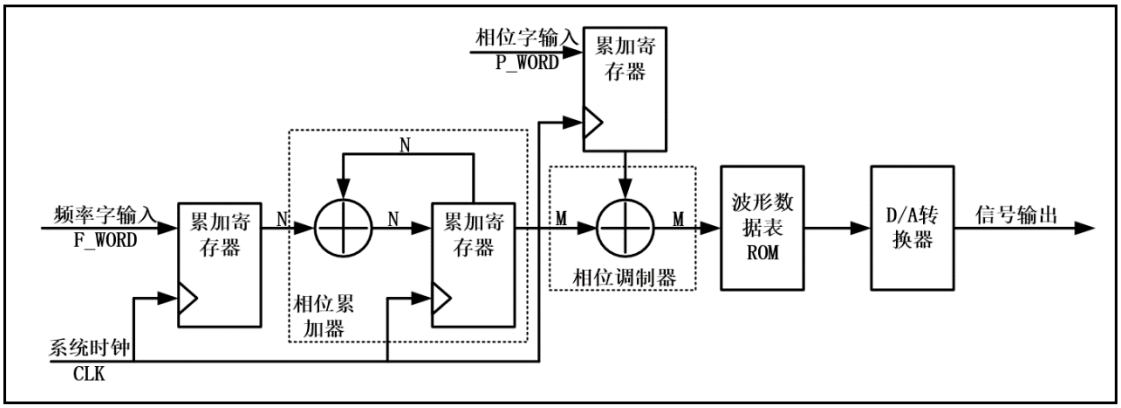

图 26-1 DDS 结构示意图（N、M 表示位宽）

图 26-1 中是不含低通滤波器的 DDS 结构示意图，图中主要包括相位累加器、相位调制器、波形存储器、数模(D/A)转换器等四大结构。接下来我们会结合 DDS 结构示意图，为大家讲解一下 DDS 的工作原理。

在讲解之前，先来对其中各参数做一下说明。系统时钟 CLK 为整个系统的工作时钟，频率为 f_{CLK}；频率字输入 F_WORD，一般为整数，数值大小控制输出信号的频率大小，数值越大输出信号频率越高，反之，输出信号频率越低，后文中用 K 表示；相位字输入 P_WORD，为整数，数值大小控制输出信号的相位偏移，主要用于相位的信号调制，后文用 P 表示；设输出信号为 CLK_OUT，频率为 f_{OUT}。

图中所展示的四大结构中，相位累加器是整个 DDS 的核心，在这里完成相位累加，生成相位码。相位累加器的输入为频率字输入 K，表示相位增量，设其位宽为 N，满足等式 $K = 2^N * f_{OUT} / f_{CLK}$ 。其在输入相位累加器之前，在系统时钟同步下做数据寄存，数据改变时不会干扰相位累加器的正常工作。

相位调制器接收相位累加器输出的相位码， 在这里加上一个相位偏移值 P，主要用于信号的相位调制，如应用于通信方面的相移键控等，不使用此部分时可以去掉，或者将其设为一个常数输入，同样相位字输入也要做寄存。

波形数据表 ROM 中存有一个完整周期的正弦波信号。假设波形数据 ROM 的地址位宽为 12 位，存储数据位宽为 8 位，即 ROM 有 $2^{12} = 4096$ 个存储空间，每个存储空间可存储 1 字节数据。将一个周期的正弦波信号，沿横轴等间隔采样 $2^{12} = 4096$ 次，每次采集的信号幅度用 1 字节数据表示，最大值为 255，最小值为 0。将 4096 次采样结果按顺序写入 ROM 的 4096 个存储单元，一个完整周期正弦波的数字幅度信号写入了波形数据表 ROM 中。波形数据表 ROM 以相位调制器传入的相位码为 ROM 读地址，将地址对应存储单元中的电压幅值数字量输出。

D/A 转换器将输入的电压幅值数字量转换为模拟量输出，就得到输出信号 CLK_OUT。

输出信号 CLK_OUT 的信号频率 $f_{OUT} = K * f_{CLK} / 2^N$。当 K = 1 时，可得 DDS 最小分辨率为：$f_{OUT} = f_{CLK} / 2^N$，此时输出信号频率最低。根据采样定理，K 的最大值应小于 $2^N / 2$。

讲到这里，读者会心存疑虑，相位累加器得到的相位码是如何实现 ROM 寻址的呢？

对于 N 位的相位累加器，它对应的相位累加值为 2^N，如果正弦 ROM 中存储单元的个数也是 2^N 的话，这个问题就很好解决，但是这对 ROM 的对存储容量的要求较高。在实际操作中，我们使用相位累加值的高几位对 ROM 进行寻址，也就是说并不是每个系统时钟都对 ROM 进行数据读取，而是多个时钟读取一次，因为这样能保证相位累加器溢出时，从正弦 ROM 表中取出正好一个正弦周期的样点。

因此，相位累加器每计数 2^N 次，对应一个正弦周期。而相位累加器 1 秒钟计数 f_{CLK} 次，在 k=1 时，DDS 输出的时钟频率就是频率分辨率。频率控制字 K 增加时，相位累加器溢出的频率增加，对应 DDS 输出信号 CLK_OUT 频率变为 K 倍的 DDS 频率分辨率。

举个例子：

设：ROM 存储单元个数为 4096，每个存储数据用 8 位二进制表示。即，ROM 地址线宽度为 12，数据线宽度为 8；相位累加器位宽 N = 32。

根据上述条件可以知道，相位调制器位宽 M = 12，那么根据 DDS 原理。那么在相位调制器中与相位控制字进行累加时，应用相位累加器的高 12 位累加。而相位累加器的低 20 位只与频率控制字累加。

我们以频率控制字 K = 1 为例，相位累加器的低 20 位一直会加 1，直到低 20 位溢出向高 12 位进位，此时 ROM 为 0，也就是说，ROM 的 0 地址中的数据被读了 2^{20} 次，继续下去，ROM 中的 4096 个点，每个点都将会被读 2^{20} 次，最终输出的波形频率应该是参考时钟

频率的 1 / 2^{20}，周期被扩大了 2^{20} 倍。同样当频率控制字为 100 时，相位累加器的低 20 位一直会加 100，那么，相位累加器的低 20 位溢出的时间比上面会快 100 倍，则 ROM 中的每个点相比于上面会少读 100 次，所以最终输出频率是上述的 10 倍。

自波形数据表 ROM 输出的波形数据传入 D/A 转换器转换为模拟信号。D/A 转换器即数/模转换器，简称 DAC（Digital to Analog Conver），是指将数字信号转换为模拟信号的电子元件或电路。

DAC 内部电路构造无太大差异，大多数 DAC 由电阻阵列和 n 个电流开关(或电压开关)构成，按照输入的数字值进行开关切换，输出对应电流或电压。因此，按照输出信号类型可分为电压型和电流型，也可以按照 DAC 能否做乘法运算进行分类。若将 DAC 分为电压型和电流型两大类，电压型 DAC 中又有权电阻网络、T 形电阻网络、树形开关网络等分别；电流型 DAC 中又有权电流型电阻网络和倒 T 形电阻网络等。

电压输出型 DAC 一般采用内置输出放大器以低阻抗输出，少部分直接通过电阻阵列进行电压输出。直接输出电压的 DAC 仅用于高阻抗负载，由于无输出放大器部分的延迟，故常作为高速 DAC 使用。

电流输出型 DAC 很少直接利用电流输出，大多外接电流 - 电压转换电路进行电压输出。实现电流 - 电压转换，方法有二：一是只在输出引脚上接负载电阻而进行电流- 电压转换，二是外接运算放大器。

DAC 的主要技术指标包括分辨率、线性度、转换精度和转换速度。

分辨率指输出模拟电压的最小增量，即表明 DAC 输入一个最低有效位(LSB)而在输出端上模拟电压的变化量。

线性度在理想情况下，DAC 的数字输入量作等量增加时，其模拟输出电压也应作等量增加，但是实际输出往往有偏离。

D/A 转换器的转换精度与 D/A 转换器的集成芯片的结构和接口电路配置有关。如果不考虑其他 D/A 转换误差时，D/A 的转换精度就是分辨率的大小，因此要获得高精度的 D/A 转换结果，首先要保证选择有足够分辨率的 D/A 转换器。同时 D/A 转换精度还与外接电路的配置有关，当外部电路器件或电源误差较大时，会造成较大的 D/A 转换误差，当这些误差超过一定程度时，D/A 转换就产生错误。

转换速度一般由建立时间决定。建立时间是将一个数字量转换为稳定模拟信号所需的时间，也可以认为是转换时间。DA 中常用建立时间来描述其速度，而不是 AD 中常用的转换速率。一般地，电流输出 DA 建立时间较短，电压输出 DA 则较长。

26.3 实战演练

26.3.1 实验目标

使用 FPGA 开发板和外部挂载的高速 AD/DA 板卡，设计并实现一个简易 DDS 信号发生器，可通过按键控制实现正弦波、方波、三角波和锯齿波的波形输出，频率相位可调。

26.3.2 硬件资源

如图 26-2 所示，我们使用外载 AD/DA 板卡的 DA 部分完成本次实验设计；如图 26-3 所示为外载 AD/DA 板卡 DA 部分原理图，包括高速 DA 芯片和外围电路。

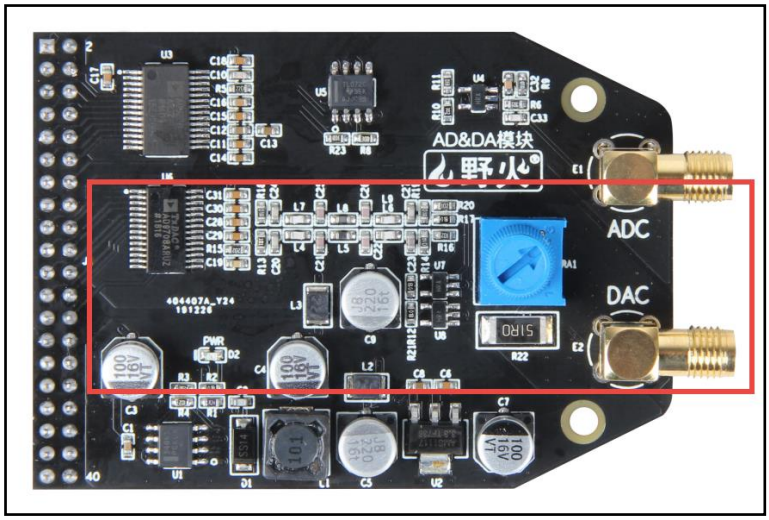

图 26-2 外载 AD/DA 板卡外观图

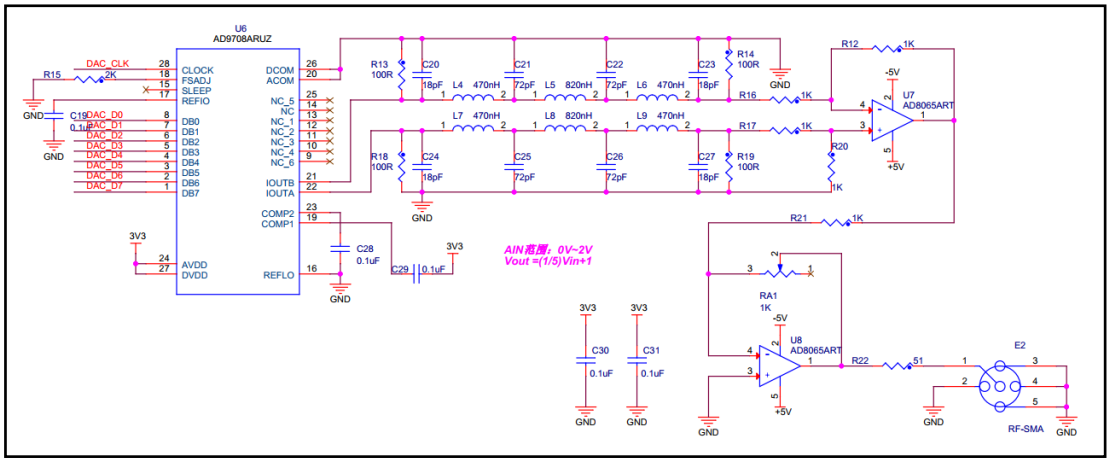

图 26-3 外载 AD/DA 板卡 DA 部分原理图

外载 AD/DA 板卡的 DA 部分使用高速 DA 芯片 AD9708，AD9708 由 ANALOG 公司生产，属于 TxDAC™系列高性能、低功耗 CMOS 数模转换器(DAC)的 8 位分辨率产品。

TxDAC™系列由引脚兼容的 8、10、12、14 位 DAC 组成，并专门针对通信系统的发射信号路径进行了优化。所有器件都采用相同的接口选项、小型封装和引脚排列，因而可以根据性能、分辨率和成本，向上或向下选择适合的器件。

AD9708 提供出色的交流和直流性能，同时支持最高 125 MSPS 的更新速率。具有灵活的单电源工作电压范围（2.7 V 至 5.5 V）和低功耗特性，非常适合便携式和低功耗应用。通过降低满量程电流输出，可以将功耗进一步降至 45 mW，而性能不会明显下降，此外，在省电模式下，待机功耗可降至约 20 mW。采用先进的 CMOS 工艺制造。分段电流源架构与专有开关技术相结合，可减小杂散分量，并增强了动态性能。该器件还集成边沿触发式

输入锁存器和一个温度补偿带隙基准电压源，可提供一个完整的单芯片 DAC 解决方案。灵活的电源选项支持+3 V 和+5 V CMOS 逻辑系列。

AD9708 是一款电流输出 DAC，标称满量程输出电流为 20 mA，输出阻抗大于 100 kΩ。它提供差分电流输出，以支持单端或差分应用。电流输出可以直接连至一个输出电阻，以提供两路互补的单端电压输出。可兼容输出电压范围为 1.25 V。内置一个 1.2 V 片内基准电压源和基准电压控制放大器，只需用单个电阻便可轻松设置满量程输出电流。该器件可以采用多种外部基准电压驱动。其满量程电流可以在 2 mA 至 20 mA 范围内调节，动态性能不受影响。因此，AD9708 能够以低功耗水平工作，或在 20 dB 范围内进行调节，进一步提供增益范围调整能力。

AD9708 采用 28 引脚 SOIC 封装，引脚图及内部结构图如下图 26-4、图 26-5 所示。关于 AD9708 的更多详细资料可查阅数据手册。

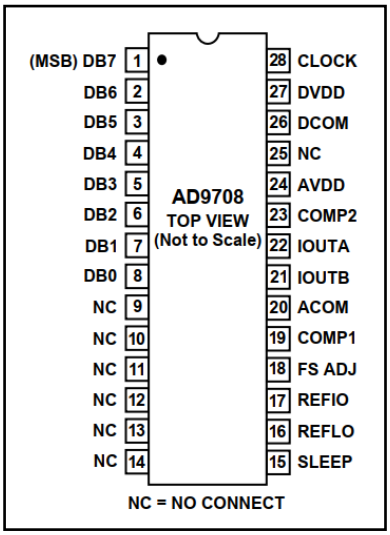

图 26-4 AD9708 引脚图

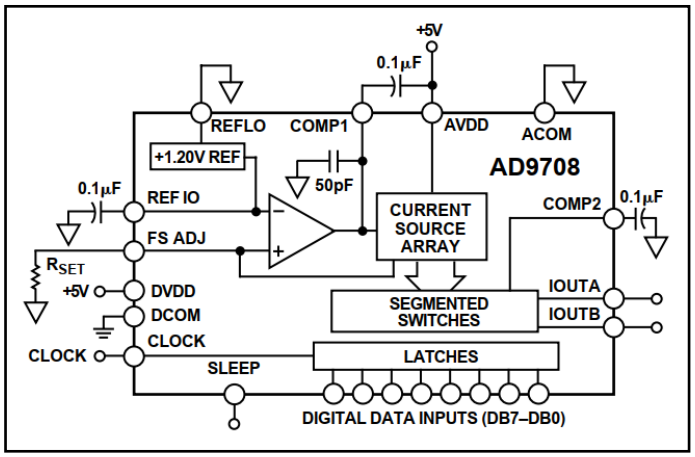

图 26-5 AD9708 内部结构图

26.3.3 程序设计

学习了 DDS 的相关知识，了解了硬件资源，根据要求进行实验工程的程序设计，接下来我们会先对实验工程进行整体说明，随后对相关子功能模块做系统讲解。

1. 整体说明

由理论知识小节可知，要实现 DDS 信号发生器需要 4 部分，D/A 转换器交由外部挂载的高速 AD/DA 板卡处理，其他 3 部分，相位累加器、相位调制器、波形数据表 ROM 由 FPGA 负责。所以我们要建立一个单独的模块对 DDS 部分进行处理；实验目标还提到要使用按键实现 4 种波形的切换，按键消抖模块必不可少；同时也要声明一个按键控制模块对 4 个输入按键进行控制，子功能模块已经足够了，最后再加一个顶层模块。

综上所述，得到实验工程整体框图如图 26-6 所示。

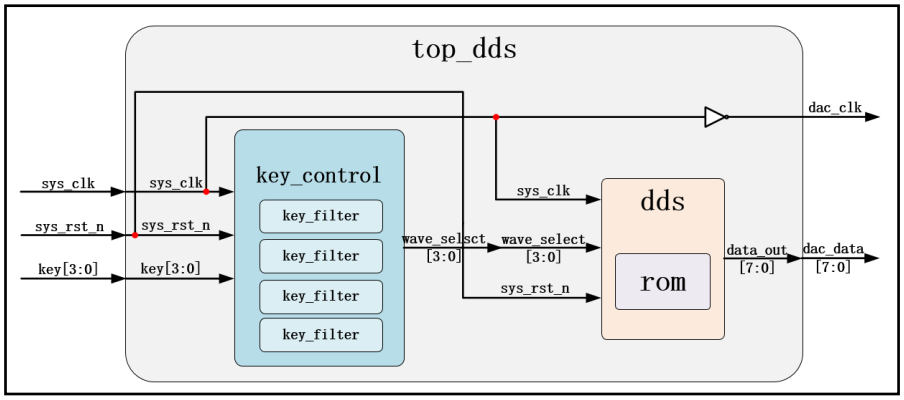

图 26-6 实验工程整体框图

时钟、复位和代表波形选择的 4 个按键信号通过顶层传入按键控制模块(key_control)，按键控制模块内部实例化 4 个按键消抖模块，对输入的 4 路按键信号分别进行消抖处理；消抖处理后的 4 路按键信号组成波形选择信号输入 dds 模块(dds)， dds 模块中实例化一个 ROM IP 核，按顺序存入了一个完整周期的正弦波、方波、三角波、锯齿波的信号波形，根据输入波形选择信号对 rom 中对应信号波形进行读取，将读出波形的幅度数字值输出，传入外部挂载的高速 AD/DA 板卡的 DA 端，板卡根据输入的数字信号生成对应波形的模拟信号。其中，输出信号的频率和相位的调节可在 dds 模块中通过修改参数实现。

2. ROM 内波形数据写入

我们设计的 DDS 简易信号发生器想要实现正弦波、方波、三角波和锯齿波 4 种波形的输出，需要事先在波形数据表 ROM 中存入 4 种波形信号各自的完整周期波形数据。ROM 作为只读存储器，在进行 IP 核设置时需要指定初始化文件，我们将波形数据作为初始化文件写入其中，文件格式为 COE 文件。

使用 MatLab 绘制 4 种信号波形，对波形进行等间隔采样，以采样次数作为 ROM 存储地址，将采集的波形幅值数据做为存储数据写入存储地址对应的存储空间。在本次实验中

我们对 4 种信号波形进行分别采样，采样次数为 2^{12} = 4096 次，采集的波形幅值数据位宽为 8bit，将采集数据保存为 MIF 文件。

各波形采样参考代码如下。

代码清单 26-1 正弦信号波形采集参考代码(sin_wave.m)

```
clc;                        %清除命令行命令
clear all;                  %清除工作区变量,释放内存空间
F1=1;                       %信号频率
Fs=2^12;                    %采样频率
P1=0;                       %信号初始相位
N=2^12;                     %采样点数
t=[0:1/Fs:(N-1)/Fs];        %采样时刻
ADC=2^7 - 1;                %直流分量
A=2^7;                      %信号幅度
%生成正弦信号
s=A*sin(2*pi*F1*t + pi*P1/180) + ADC;
plot(s);                    %绘制图形
%创建 coe 文件
fild = fopen('sin_wave_4096x8.coe','wt');
%写入 coe 文件头
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_RADIX=10;'); %10 进制数
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_VECTOR=');
for i = 1:N
    s0(i) = round(s(i));     %对小数四舍五入以取整
    if s0(i) <0              %负 1 强制置零
        s0(i) = 0
    end
    if i == N
            fprintf(fild, '%d',s0(i));        %数据写入
            fprintf(fild, '%s',';');          %最后一个数据使用分号结束
    else
            fprintf(fild, '%d',s0(i));        %数据写入
            fprintf(fild, '%s\n',',');        %逗号，换行
    end
end
fclose(fild);
```

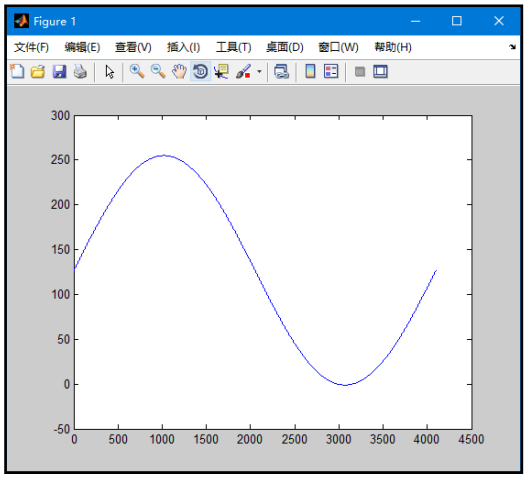

图 26-7 MatLab 生成的正弦信号波形

代码清单 26-2 方波信号波形采集参考代码(squ_wave.m)

```
clc;                        %清除命令行命令
clear all;                  %清除工作区变量,释放内存空间
```

```
F1=1;                       %信号频率
Fs=2^12;                    %采样频率
P1=0;                       %信号初始相位
N=2^12;                     %采样点数
t=[0:1/Fs:(N-1)/Fs];        %采样时刻
ADC=2^7 - 1;                %直流分量
A=2^7;                      %信号幅度
%生成方波信号
s=A*square(2*pi*F1*t + pi*P1/180) + ADC;
plot(s);                    %绘制图形
%创建 coe 文件
fild = fopen('squ_wave_4096x8.coe','wt');
%写入 coe 文件头
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_RADIX=10;'); %10 进制数
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_VECTOR=');
for i = 1:N
    s0(i) = round(s(i));     %对小数四舍五入以取整
    if s0(i) <0              %负 1 强制置零
        s0(i) = 0
    end
    if i == N
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s',';');         %最后一个数据使用分号结束
    else
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s\n',',');       %逗号，换行
    end
end
fclose(fild);
```

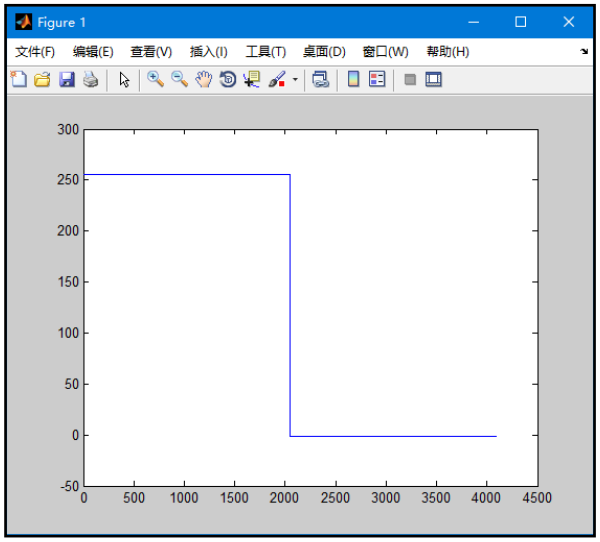

图 26-8 MatLab 生成的方波信号波形

代码清单 26-3 三角波信号波形采集参考代码(tri_wave.m)

```
clc;                        %清除命令行命令
clear all;                  %清除工作区变量,释放内存空间
F1=1;                       %信号频率
Fs=2^12;                    %采样频率
P1=0;                       %信号初始相位
N=2^12;                     %采样点数
t=[0:1/Fs:(N-1)/Fs];        %采样时刻
```

```
ADC=2^7 - 1;                %直流分量
A=2^7;                      %信号幅度
%生成三角波信号
s=A*sawtooth(2*pi*F1*t + pi*P1/180,0.5) + ADC;
plot(s);                    %绘制图形
%创建 coe 文件
fild = fopen('tri_wave_4096x8.coe','wt');
%写入 coe 文件头
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_RADIX=10;'); %10 进制数
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_VECTOR=');
for i = 1:N
    s0(i) = round(s(i));     %对小数四舍五入以取整
    if s0(i) <0              %负 1 强制置零
        s0(i) = 0
    end
    if i == N
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s',';');         %最后一个数据使用分号结束
    else
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s\n',',');       %逗号，换行
    end
end
fclose(fild);
```

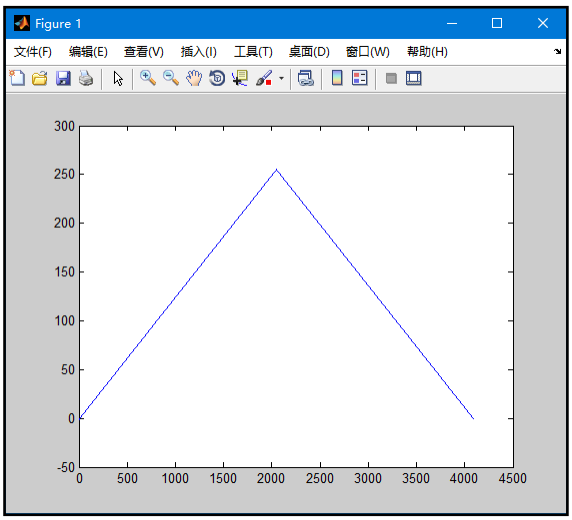

图 26-9 MatLab 生成的三角波信号波形

代码清单 26-4 锯齿波信号波形采集参考代码(saw_wave.m)

```
clc;                        %清除命令行命令
clear all;                  %清除工作区变量,释放内存空间
F1=1;                       %信号频率
Fs=2^12;                    %采样频率
P1=0;                       %信号初始相位
N=2^12;                     %采样点数
t=[0:1/Fs:(N-1)/Fs];        %采样时刻
ADC=2^7 - 1;                %直流分量
A=2^7;                      %信号幅度
%生成锯齿波信号
s=A*sawtooth(2*pi*F1*t + pi*P1/180) + ADC;
plot(s);                    %绘制图形
%创建 coe 文件
fild = fopen('saw_wave_4096x8.coe','wt');
```

```
%写入 coe 文件头
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_RADIX=10;'); %10 进制数
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_VECTOR=');
for i = 1:N
    s0(i) = round(s(i));     %对小数四舍五入以取整
    if s0(i) <0              %负 1 强制置零
        s0(i) = 0
    end
    if i == N
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s',';');         %最后一个数据使用分号结束
    else
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s\n',',');       %逗号，换行
    end
end
fclose(fild);
```

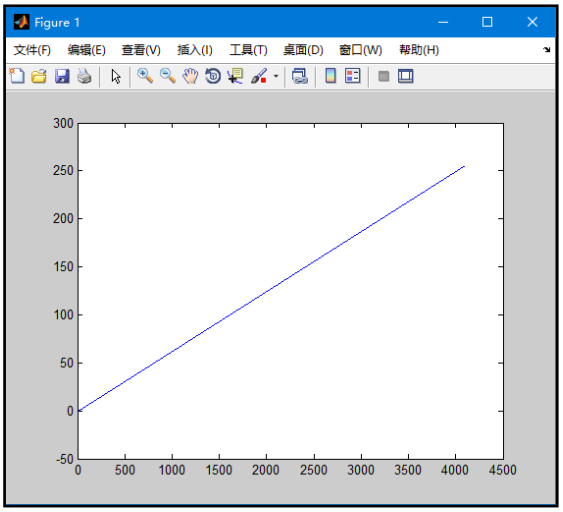

图 26-10 MatLab 生成的锯齿波信号波形

使用 MatLab 对 4 种波形进行采样后，生成 4 个 MIF 文件，分别对应 4 种波形，我们可以调用 4 个深度为 4096，位宽为 8bit 的 ROM IP 核，将 4 种波形对应的 COE 文件分别写入，生成 4 个波形数据表 ROM。

也可以调用一个深度为 4096*4，位宽为 8bit 的 ROM，生成一个 COE 文件，这个 COE 文件位上述 4 个 COE 文件的集合波形数据按照正弦波、方波、三角波、锯齿波的顺序写入。总的 COE 文件生成代码如下所示。

代码清单 26-5 整体信号波形采集参考代码(wave_16384x8.m)

```
clc;                        %清除命令行命令
clear all;                  %清除工作区变量,释放内存空间
F1=1;                       %信号频率
Fs=2^12;                    %采样频率
P1=0;                       %信号初始相位
N=2^12;                     %采样点数
t=[0:1/Fs:(N-1)/Fs];        %采样时刻
ADC=2^7 - 1;                %直流分量
A=2^7;                      %信号幅度
s1=A*sin(2*pi*F1*t + pi*P1/180) + ADC;          %正弦波信号
s2=A*square(2*pi*F1*t + pi*P1/180) + ADC;       %方波信号
```

```
s3=A*sawtooth(2*pi*F1*t + pi*P1/180,0.5) + ADC; %三角波信号
s4=A*sawtooth(2*pi*F1*t + pi*P1/180) + ADC;      %锯齿波信号
%创建 coe 文件
fild = fopen('wave_16384x8.coe','wt');
%写入 coe 文件头
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_RADIX=10;'); %10 进制数
fprintf(fild, '%s\n','MEMORY_INITIALIZATION_VECTOR=');
for j = 1:4
    for i = 1:N
        if j == 1        %打印正弦信号数据
            s0(i) = round(s1(i));     %对小数四舍五入以取整
        end

        if j == 2        %打印方波信号数据
            s0(i) = round(s2(i));     %对小数四舍五入以取整
        end

        if j == 3        %打印三角波信号数据
            s0(i) = round(s3(i));     %对小数四舍五入以取整
        end

        if j == 4        %打印锯齿波信号数据
            s0(i) = round(s4(i));     %对小数四舍五入以取整
        end

        if s0(i) <0              %负 1 强制置零
            s0(i) = 0
        end

        if j == 4 && i == N
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s',';');         %最后一个数使用分号结束
        else
            fprintf(fild, '%d',s0(i));       %数据写入
            fprintf(fild, '%s\n',',');       %逗号，换行
        end
    end
end
fclose(fild);
```

本次实验采用第二种方法，生成了一个波形数据表 ROM，将 4 种信号波形数据按照正弦波、方波、三角波、锯齿波的顺序写入。

3. 按键控制模块

模块设计

本实验设计的 DDS 信号发生器，可以实现 4 种信号波形的输出，使用外部物理按键实现波形的切换，一个按键控制一种波形，共使用 4 个按键。外部物理按键的触发信号通过顶层模块输入按键控制模块，按键控制模块内部实例化 4 个按键消抖消抖模块，分别对 4 路按键信号做消抖处理。消抖处理后的 4 路按键信号组成位宽为 4bit 的波形选择信号并输出至 DDS 模块。波形选择信号初值为 4'b0000，当某一按键按下，波形选择信号对应位电平拉高。模块框图，具体见图 26-11。

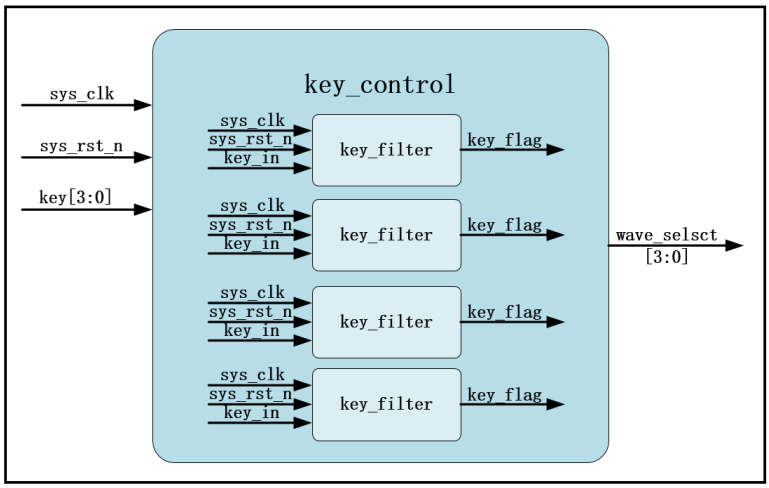

图 26-11 按键控制模块框图

波形图绘制

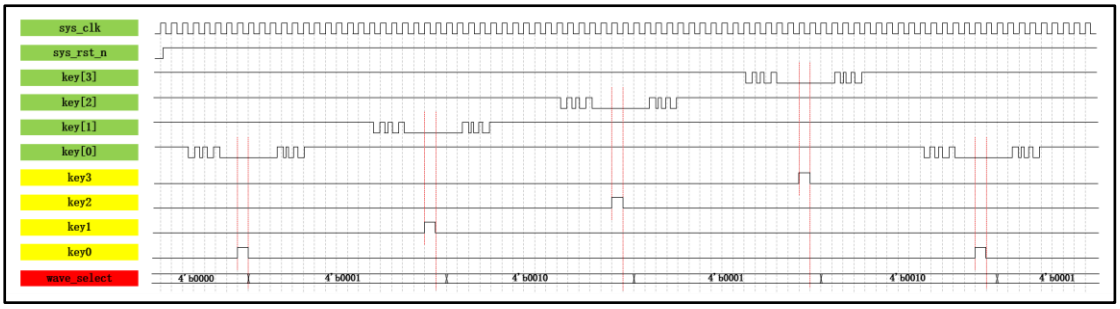

图 26-12 按键控制模块整体波形图

由图可知，输入的信号有 3 路，时钟、复位和未消抖的按键控制，为了方便观察，将未消抖的按键控制信号拆开绘制，每一位按键信号代表一种信号波形的选择，但有一点要注意，每次只能按下一个按键。

首先要对 4 路未消抖的按键信号进行消抖处理，这里需要调用 4 个按键消抖模块，如图 26-11 所示。消抖完毕之后我们需要声明 4 个变量将消抖后的按键信号引出，即波形图中的黄色变量 key0、key1、key2、key3。

接下来，使用引出的消抖后的按键信号 key0、key1、key2、key3 为约束条件，产生我们的波形选择信号 wave_select，给他一个初值 4'b0000，当按键信号 key0 为高电平，wave_select 赋值为 4'b0001，表示输出波形为正弦波；当按键信号 key1、key2、key3 各自为高电平时，wave_select 分别赋值为 4'b0010、4'b0100、4'b1000，表示输出波形分别为方波、三角波、锯齿波，其他状态保持原值不变。

代码编写

参照波形图编写模块参考代码，参考代码如代码清单 26-6 所示。

代码清单 26-6 按键控制模块参考代码(key_control.v)

```
001 module  key_control
002 (
```

```
003     input   wire            sys_clk     ,   //系统时钟,50MHz
004     input   wire            sys_rst_n   ,   //复位信号,低电平有效
005     input   wire    [3:0]   key         ,   //输入 4 位按键
006
007     output  reg     [3:0]   wave_select     //输出波形选择
008 );
009
010 //*****************************************************************//
011 //****************** Parameter and Internal Signal ******************//
012 //*****************************************************************//
013 //parameter define
014 parameter   sin_wave    =   4'b0001,    //正弦波
015             squ_wave    =   4'b0010,    //方波
016             tri_wave    =   4'b0100,    //三角波
017             saw_wave    =   4'b1000;    //锯齿波
018
019 parameter   CNT_MAX =   20'd999_999;    //计数器计数最大值
020
021 //wire  define
022 wire            key3    ;   //按键 3
023 wire            key2    ;   //按键 2
024 wire            key1    ;   //按键 1
025 wire            key0    ;   //按键 0
026
027 //*****************************************************************//
028 //***************************** Main Code *****************************//
029 //*****************************************************************//
030 //wave:按键状态对应波形
031 always@(posedge sys_clk or negedge sys_rst_n)
032     if(sys_rst_n == 1'b0)
033         wave_select   <=  4'b0000;
034     else    if(key0 == 1'b1)
035         wave_select   <=  sin_wave;
036     else    if(key1 == 1'b1)
037         wave_select   <=  squ_wave;
038     else    if(key2 == 1'b1)
039         wave_select   <=  tri_wave;
040     else    if(key3 == 1'b1)
041         wave_select   <=  saw_wave;
042     else
043         wave_select   <=  wave_select;
044
045 //*****************************************************************//
046 //************************** Instantiation **************************//
047 //*****************************************************************//
048 //------------- key_fifter_inst3 --------------
049 key_filter
050 #(
051     .CNT_MAX      (CNT_MAX  )       //计数器计数最大值
052 )
053 key_filter_inst3
054 (
055     .sys_clk      (sys_clk  )   ,   //系统时钟 50Mhz
056     .sys_rst_n    (sys_rst_n)   ,   //全局复位
057     .key_in       (key[3]   )   ,   //按键输入信号
058
059     .key_flag     (key3     )       //按键消抖后标志信号
060 );
061
062 //------------- key_fifter_inst2 --------------
063 key_filter
064 #(
065     .CNT_MAX      (CNT_MAX  )       //计数器计数最大值
```

```
066 )
067 key_filter_inst2
068 (
069     .sys_clk      (sys_clk  )    ,    //系统时钟 50Mhz
070     .sys_rst_n    (sys_rst_n)    ,    //全局复位
071     .key_in       (key[2]   )    ,    //按键输入信号
072
073     .key_flag     (key2     )         //按键消抖后标志信号
074 );
075
076 //------------- key_fifter_inst1 --------------
077 key_filter
078 #(
079     .CNT_MAX      (CNT_MAX  )         //计数器计数最大值
080 )
081 key_filter_inst1
082 (
083     .sys_clk      (sys_clk  )    ,    //系统时钟 50Mhz
084     .sys_rst_n    (sys_rst_n)    ,    //全局复位
085     .key_in       (key[1]   )    ,    //按键输入信号
086
087     .key_flag     (key1     )         //按键消抖后标志信号
088 );
089
090 //------------- key_fifter_inst0 --------------
091 key_filter
092 #(
093     .CNT_MAX      (CNT_MAX  )         //计数器计数最大值
094 )
095 key_filter_inst0
096 (
097     .sys_clk      (sys_clk  )    ,    //系统时钟 50Mhz
098     .sys_rst_n    (sys_rst_n)    ,    //全局复位
099     .key_in       (key[0]   )    ,    //按键输入信号
100
101     .key_flag     (key0     )         //按键消抖后标志信号
102 );
103
104 endmodule
```

仿真验证

仿真文件编写

模块代码编写完成，编写仿真文件对模块代码进行仿真验证，仿真文件参考代码如下代码清单 26-7 所示。

代码清单 26-7 按键控制模块仿真参考代码(tb_key_control.v)

```
01 `timescale  1ns/1ns
02
03 module  tb_key_control();
04
05 //****************************************************************//
06 //*************** Parameter and Internal Signal ***************//
07 //****************************************************************//
08 parameter   CNT_1MS  = 20'd19   ,
09            CNT_11MS = 21'd69   ,
10            CNT_41MS = 22'd149  ,
11            CNT_51MS = 22'd199  ,
12            CNT_60MS = 22'd249  ;
13
14 //wire  define
```

```
15 wire    [3:0]   wave_select ;
16
17 //reg   define
18 reg             sys_clk     ;
19 reg             sys_rst_n   ;
20 reg     [21:0]  tb_cnt      ;
21 reg             key_in      ;
22 reg     [1:0]   cnt_key     ;
23 reg     [3:0]   key         ;
24
25 //defparam  define
26 defparam    key_control_inst.CNT_MAX = 24;
27
28 //*************************************************************//
29 //************************* Main Code *************************//
30 //*************************************************************//
31 //sys_rst_n,sys_clk,key
32 initial
33     begin
34         sys_clk     =   1'b0;
35         sys_rst_n  <=   1'b0;
36         key <= 4'b0000;
37         #200;
38         sys_rst_n   <=   1'b1;
39     end
40
41 always #10 sys_clk = ~sys_clk;
42
43 //tb_cnt:按键过程计数器，通过该计数器的计数时间来模拟按键的抖动过程
44 always@(posedge sys_clk or negedge sys_rst_n)
45     if(sys_rst_n == 1'b0)
46         tb_cnt <= 22'b0;
47     else    if(tb_cnt == CNT_60MS)
48         tb_cnt <= 22'b0;
49     else
50         tb_cnt <= tb_cnt + 1'b1;
51
52 //key_in:产生输入随机数，模拟按键的输入情况
53 always@(posedge sys_clk or negedge sys_rst_n)
54     if(sys_rst_n == 1'b0)
55         key_in <= 1'b1;
56     else    if((tb_cnt >= CNT_1MS && tb_cnt <= CNT_11MS)
57                 || (tb_cnt >= CNT_41MS && tb_cnt <= CNT_51MS))
58         key_in <= {$random} % 2;
59     else    if(tb_cnt >= CNT_11MS && tb_cnt <= CNT_41MS)
60         key_in <= 1'b0;
61     else
62         key_in <= 1'b1;
63
64 always@(posedge sys_clk or negedge sys_rst_n)
65     if(sys_rst_n == 1'b0)
66         cnt_key <=  2'd0;
67     else    if(tb_cnt == CNT_60MS)
68         cnt_key <=  cnt_key + 1'b1;
69     else
70         cnt_key <=  cnt_key;
71
72 always@(posedge sys_clk or negedge sys_rst_n)
73     if(sys_rst_n == 1'b0)
74         key     <=  4'b1111;
75     else
76         case(cnt_key)
77             0:      key <=  {3'b111,key_in};
78             1:      key <=  {2'b11,key_in,1'b1};
79             2:      key <=  {1'b1,key_in,2'b11};
80             3:      key <=  {key_in,3'b111};
```

```
81              default:key <=  4'b1111;
82          endcase
83
84  //*************************************************************//
85  //*********************** Instantiation ***********************//
86  //*************************************************************//
87  //------------- key_control_inst -------------
88  key_control     key_control_inst
89  (
90      .sys_clk      (sys_clk    ),    //系统时钟,50MHz
91      .sys_rst_n    (sys_rst_n  ),    //复位信号,低电平有效
92      .key          (key        ),    //输入 4 位按键
93
94      .wave_select (wave_select)     //输出波形选择
95  );
96
97  endmodule
```

仿真波形分析

仿真参考代码编写完成，使用 Modelsim 对按键控制模块进行仿真验证，仿真波形如下图 26-13、图 26-14 所示。由图可知，各信号仿真波形与绘制波形图波形变化一致，模块仿真验证通过。

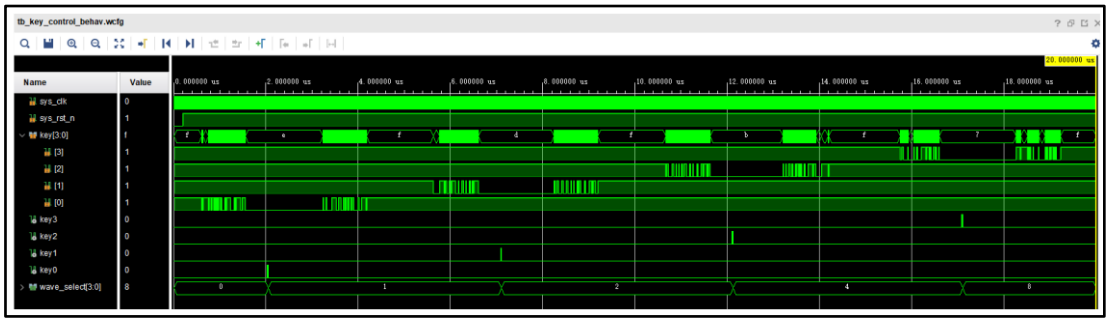

图 26-13 按键控制模块整体仿真波形

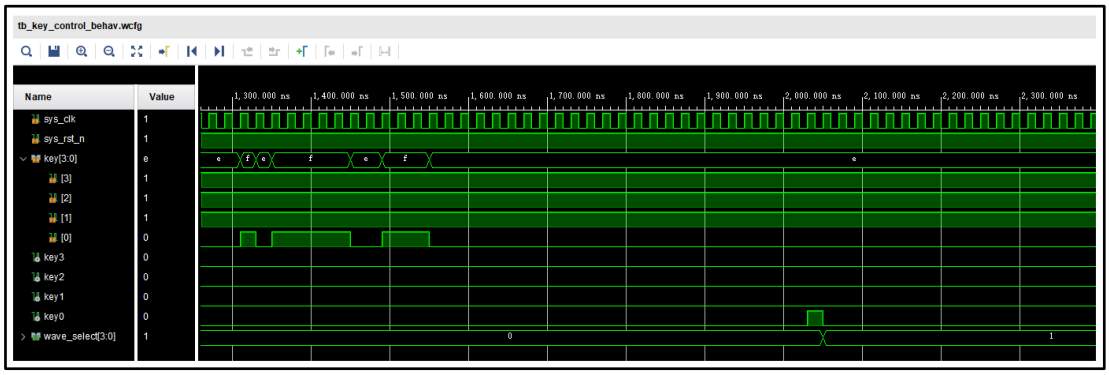

图 26-14 按键控制模块局部仿真波形

4. DDS 模块

模块框图

波形数据表 ROM 生成完毕，按键控制模块也做了说明，我们开始本实验工程的核心模块 DDS 模块的讲解。DDS 模块框图，具体见图 26-15。

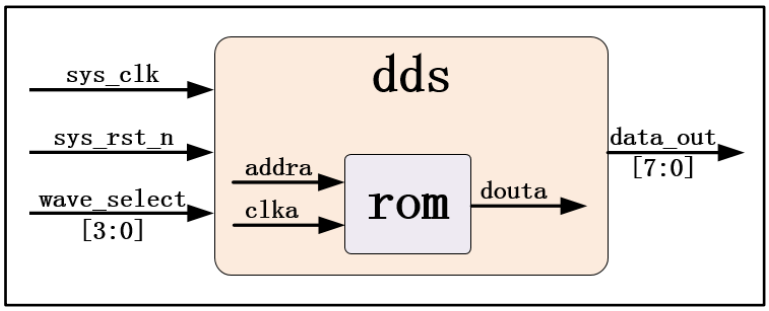

图 26-15 DDS 模块框图

DDS 模块有 3 路输入信号，1 路输出信号，其内部例化了前面生成的波形数据表 ROM；输入信号中有时钟 sys_clk、复位 sys_rst_n 和按键控制模块输入的波形选择信号 wave_select。输入的波形选择信号有 4 种状态，分别对应 4 中波形，根据输入的波形选择信号的不同，对 ROM 中波形选择信号对应波形的存储位置进行数据读取，将读出波形幅值数据通过输出信号 data_out 输出到外部挂载 DA 板块，进行数模转换。

波形图绘制

DDS 模块的各输入输出信号和模块功能讲解完毕，我们开始模块波形图的绘制，对波形图各信号进行详细说明，讲解一下模块功能实现方法。DDS 模块整体波形图，具体见图 26-16。

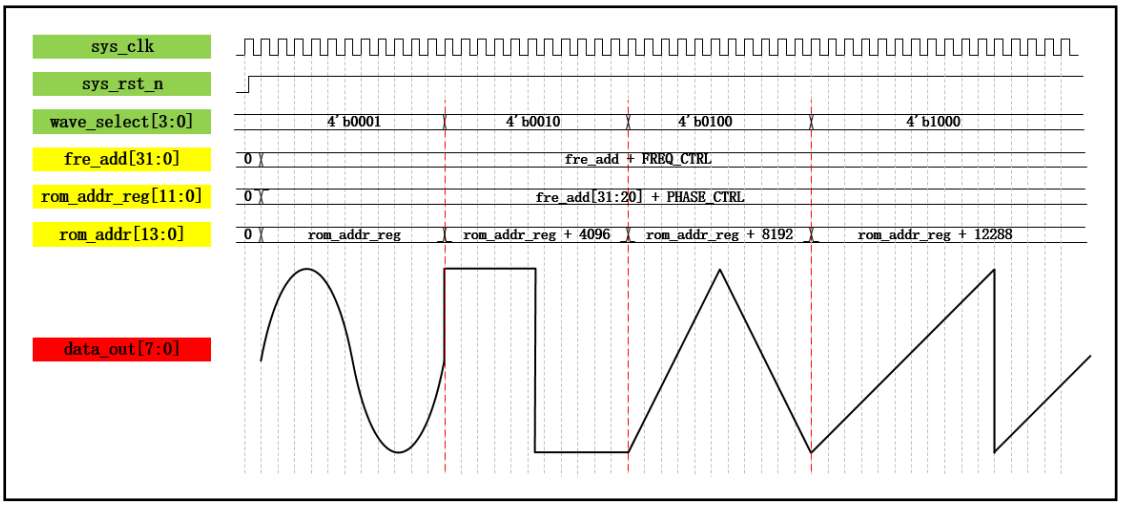

图 26-16 DDS 模块波形图

本模块包含 3 路输入信号，1 路输出信号，内部声明 3 个寄存器变量。输入信号包括时钟、复位和输出波形选择信号 wave_select。波形选择信号有 4 个状态，对应 4 种波形。

内部声明 3 个寄存器变量。其中 fre_add 表示相位累加器输出值，位宽为 32 位，系统上电后，fre_add 信号一直执行自加操作，每个时钟周期自加参数 FREQ_CTRL，参数 FREQ_CTRL 就是在之前理论知识部分提到的频率字输入 K，它的具体数值可通过公式计算得到，前面我们也讲到过。

寄存器变量 rom_addr_reg 表示相位调制器输出值，将相位累加器输出值的高 12 位与相位偏移量 PHASE_CTRL 相加，参数 PHASE_CTRL 就是我们之前提到过的相位字输入

P。之所以使用高 12 位，与存储波形的 ROM 深度有关。按理论讲，将得到的变量 rom_addr_reg，可直接作为 ROM 读地址输入波形数据表进行数据读取，但是我们将 4 中波形存储在了同一 ROM 中，所以还需要对读数据地址做进一步计算。

ROM 读地址 rom_addr 是输入波形数据表的 ROM 读地址，是在 rom_addr_reg 的基础上计算得到。我们之前将 4 种信号波形数据按照正弦波、方波、三角波、锯齿波的顺序写入 ROM。若需要读取正弦波波形数据，rom_addr_reg 可直接赋值给 rom_addr；但是要进行方波波形数据的读取，rom_addr_reg 需要再加上正弦波存储单元个数才能赋值给 rom_addr；剩余两信号同理。

对于参数 FREQ_CTRL 和 PHASE_CTRL，我们可以修改其参数值，实现不同频率、不同初相位波形的输出。

本实验，我们希望输出一个频率为 500Hz，初相位为 π /2 的正弦波信号。

计算参数 FREQ_CTRL，即频率输入字 K。

FREQ_CTRL = K = $2^N * f_{OUT} / f_{CLK}$，其中 N = 32(相位累加器输出值 fre_add 的位宽)、f_{OUT} = 500Hz，f_{CLK} = 50MHz，带入公式，FREQ_CTRL = K = 42949.67296 ，取整数部分为 42949；

计算参数 PHASE_CTRL，即相位输入字 P。

PHASE_CTRL = P = θ / ($2 π / 2^M$)，其中 M =12(输入 ROM 地址位宽)、θ = π / 2，带入公式，PHASE_CTRL = P = 1024。

ROM 读地址 rom_addr 写入波形数据表 ROM 中，读出地址对应波形波形数据，通过输出端口 data_out 输入到外部挂载板卡数模转换部分。

代码编写

参照绘制波形图，编写模块参考代码如下代码清单 26-8 所示。

代码清单 26-8 DDS 模块参考代码(dds.v)

```
01 module  dds
02 (
03     input   wire            sys_clk     ,   //系统时钟,50MHz
04     input   wire            sys_rst_n   ,   //复位信号,低电平有效
05     input   wire    [3:0]   wave_select ,   //输出波形选择
06
07     output  wire    [7:0]   data_out        //波形输出
08 );
09
10 //*************************************************************//
11 //***************** Parameter and Internal Signal *****************//
12 //*************************************************************//
13 //parameter define
14 parameter   sin_wave    =   4'b0001     ,   //正弦波
15             squ_wave    =   4'b0010     ,   //方波
16             tri_wave    =   4'b0100     ,   //三角波
17             saw_wave    =   4'b1000     ;   //锯齿波
18 parameter   FREQ_CTRL   =   32'd42949   ,   //相位累加器单次累加值
19             PHASE_CTRL  =   12'd1024    ;   //相位偏移量
20
21 //reg   define
22 reg     [31:0]  fre_add     ;   //相位累加器
23 reg     [11:0]  rom_addr_reg;   //相位调制后的相位码
```

```
24 reg     [13:0]  rom_addr    ;   //ROM 读地址
25
26 //////////////////////////////////////////////////////////////////////
27 //////////////////////////// Main Code ///////////////////////////////
28 //////////////////////////////////////////////////////////////////////
29 //fre_add:相位累加器
30 always@(posedge sys_clk or negedge sys_rst_n)
31     if(sys_rst_n == 1'b0)
32         fre_add <=  32'd0;
33     else
34         fre_add <=  fre_add + FREQ_CTRL;
35
36 //rom_addr:ROM 读地址
37 always@(posedge sys_clk or negedge sys_rst_n)
38     if(sys_rst_n == 1'b0)
39         begin
40             rom_addr        <=  14'd0;
41             rom_addr_reg    <=  11'd0;
42         end
43     else
44     case(wave_select)
45         sin_wave:
46             begin
47                 rom_addr_reg    <=  fre_add[31:20] + PHASE_CTRL;
48                 rom_addr        <=  rom_addr_reg;
49             end     //正弦波
50         squ_wave:
51             begin
52                 rom_addr_reg    <=  fre_add[31:20] + PHASE_CTRL;
53                 rom_addr        <=  rom_addr_reg + 14'd4096;
54             end     //方波
55         tri_wave:
56             begin
57                 rom_addr_reg    <=  fre_add[31:20] + PHASE_CTRL;
58                 rom_addr        <=  rom_addr_reg + 14'd8192;
59             end     //三角波
60         saw_wave:
61         begin
62                 rom_addr_reg    <=  fre_add[31:20] + PHASE_CTRL;
63                 rom_addr        <=  rom_addr_reg + 14'd12288;
64             end     //锯齿波
65         default:
66             begin
67                 rom_addr_reg    <=  fre_add[31:20] + PHASE_CTRL;
68                 rom_addr        <=  rom_addr_reg;
69             end     //正弦波
70     endcase
71
72 //////////////////////////////////////////////////////////////////////
73 //////////////////////////// Instantiation ///////////////////////////
74 //////////////////////////////////////////////////////////////////////
75 //------------------------- rom_wave_inst ------------------------
76 rom_wave    rom_wave_inst
77 (
78     .clka    (sys_clk   ),  //读时钟
79     .addra   (rom_addr  ),  //ROM 读地址
80
81     .douta   (data_out   )  //读出波形数据
82 );
83
84 endmodule
```

参考代码编写完成，其中各信号相关知识在波形图绘制小节已经做了详细讲解，此处不再赘述。

仿真验证

仿真文件编写

模块代码编写完成，编写仿真文件对模块代码进行仿真验证，仿真文件参考代码如下代码清单 26-9 所示。

代码清单 26-9 DDS 模块仿真参考代码(tb_dds.v)

```
01 `timescale  1ns/1ns
02
03 module  tb_dds();
04
05 //****************************************************************//
06 //*************** Parameter and Internal Signal ***************//
07 //****************************************************************//
08 //wire  define
09 wire    [7:0]   data_out    ;
10
11 //reg   define
12 reg             sys_clk     ;
13 reg             sys_rst_n   ;
14 reg     [3:0]   wave_select ;
15
16 //****************************************************************//
17 //************************** Main Code ***************************//
18 //****************************************************************//
19 //sys_rst_n,sys_clk,key
20 initial
21     begin
22         sys_clk     =  1'b0;
23         sys_rst_n   <=   1'b0;
24         wave_select <=  4'b0000;
25         #200;
26         sys_rst_n   <=   1'b1;
27         #10000
28         wave_select <=  4'b0001;
29         #8000000;
30         wave_select <=  4'b0010;
31         #8000000;
32         wave_select <=  4'b0100;
33         #8000000;
34         wave_select <=  4'b1000;
35         #8000000;
36         wave_select <=  4'b0000;
37         #8000000;
38     end
39
40 always #10 sys_clk = ~sys_clk;
41
42 //****************************************************************//
43 //************************ Instantiation *************************//
44 //****************************************************************//
45 //------------- top_dds_inst -------------
46 dds     dds_inst
47 (
48     .sys_clk     (sys_clk    ),    //系统时钟,50MHz
49     .sys_rst_n   (sys_rst_n  ),    //复位信号,低电平有效
50     .wave_select (wave_select),    //输出波形选择
51
52     .data_out    (data_out   )     //波形输出
53 );
```

```
54
55 endmodule
```

仿真波形分析

仿真参考代码编写完成，使用 Modelsim 对 DDS 模块进行仿真验证，仿真波形如下图 26-17 所示。由图可知，各信号仿真波形与绘制波形图波形变化一致，模块仿真验证通过。

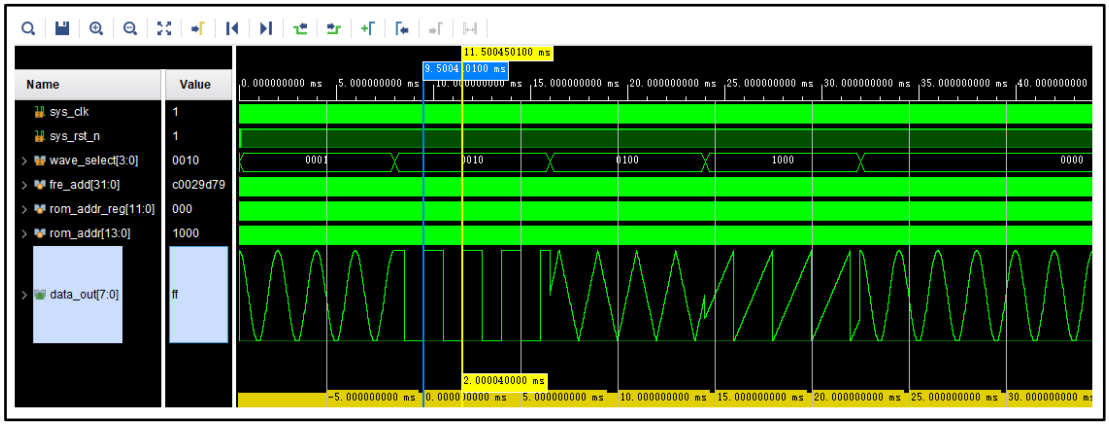

图 26-17 DDS 模块整体仿真波形

DDS 模块仿真波形图中，两调波形参考线之间的时间间隔约为 200_0000ns，表示一个方波周期，频率约为 500Hz；将仿真信号波形与 MatLab 生成波形对比，可以发现信号波形相位平移了 π/2，模块通过仿真验证。

5. 顶层模块

模块框图

顶层模块较为简单，内部例化了各子功能模块，连接各对应信号；外部有 3 路输入信号、2 路输出信号。输入有时钟、复位信号和控制信号波形切换的 4 路按键信号；输出 2 路信号中，信号 dac_data 为 DDS 模块输出的，自波形数据表 ROM 中读取的波形数据；信号 dac_clk 为输入至外载板卡的时钟信号，DA 模块使用此时钟进行数据处理，该信号由系统时钟 sys_clk 取反得到。

波形数据表 ROM 的读时钟为系统时钟 sys_clk，在系统时钟上升沿时对 ROM 进行数据读取，而 DA 模块也使用时钟上升沿进行数据处理，将系统时钟 sys_clk 取反得到 dac_clk，dac_clk 的上升沿刚好采集到波形数据 dac_data 的稳定数据。

顶层模块框图，具体见图 26-18。

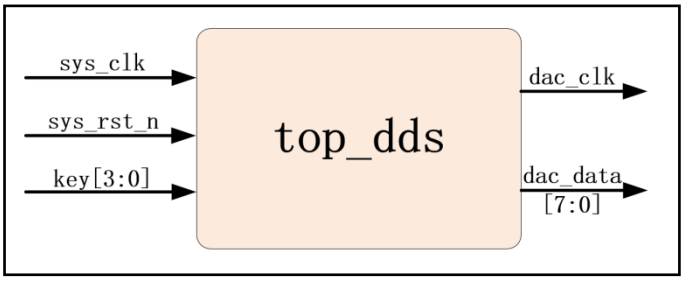

图 26-18 顶层模块框图

代码编写

对于顶层模块，我们不需要进行波形图的绘制，直接编写代码。模块参考代码，具体见代码清单 26-10。

代码清单 26-10 顶层模块参考代码(top_dds.v)

```
01 module  top_dds
02 (
03     input   wire            sys_clk     ,   //系统时钟,50MHz
04     input   wire            sys_rst_n   ,   //复位信号,低电平有效
05     input   wire    [3:0]   key         ,   //输入 4 位按键
06
07     output  wire            dac_clk     ,   //输入 DAC 模块时钟
08     output  wire    [7:0]   dac_data        //输入 DAC 模块波形数据
09 );
10
11 //*****************************************************************//
12 //****************** Parameter and Internal Signal ******************//
13 //*****************************************************************//
14 //wire  define
15 wire    [3:0]   wave_select ;   //波形选择
16
17 //dac_clka:DAC 模块时钟
18 assign  dac_clk  = ~sys_clk;
19
20 //*****************************************************************//
21 //*************************** Instantiation ***************************//
22 //*****************************************************************//
23 //--------------------------- dds_inst -----------------------------
24 dds     dds_inst
25 (
26     .sys_clk        (sys_clk    ),   //系统时钟,50MHz
27     .sys_rst_n      (sys_rst_n  ),   //复位信号,低电平有效
28     .wave_select    (wave_select),   //输出波形选择
29
30     .data_out       (dac_data   )    //波形输出
31 );
32
33 //----------------------- key_control_inst -----------------------
34 key_control key_control_inst
35 (
36     .sys_clk        (sys_clk    ),   //系统时钟,50MHz
37     .sys_rst_n      (sys_rst_n  ),   //复位信号,低电平有效
38     .key            (key        ),   //输入 4 位按键
39
40     .wave_select    (wave_select)    //输出波形选择
41  );
42
43 endmodule
```

6. RTL 原理图

顶层模块介绍完毕，使用 Vivado 软件对实验工程进行编译，工程通过编译后查看实验工程 RTL 原理图。工程 RTL 原理图，具体见图 26-19。由图可知，实验工程的 RTL 原理与实验整体框图相同，各信号线均已正确连接。

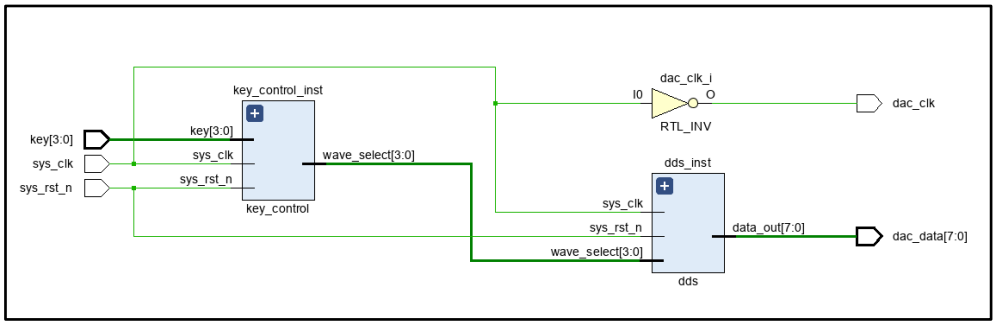

图 26-19 RTL 原理图

7. 仿真验证

仿真代码编写

编写仿真代码，对工程进行整体仿真。顶层模块仿真参考代码，具体见代码清单 26-11。

代码清单 26-11 顶层模块仿真参考代码(tb_top_dds.v)

```
01 `timescale  1ns/1ns
02
03 module  tb_top_dds();
04
05 //****************************************************************//
06 //*************** Parameter and Internal Signal ***************//
07 //****************************************************************//
08 parameter   CNT_1MS  = 20'd19000   ,
09            CNT_11MS = 21'd69000   ,
10            CNT_41MS = 22'd149000  ,
11            CNT_51MS = 22'd199000  ,
12            CNT_60MS = 22'd249000  ;
13
14 //wire  define
15 wire            dac_clk     ;
16 wire    [7:0]   dac_data    ;
17
18 //reg   define
19 reg             sys_clk     ;
20 reg             sys_rst_n   ;
21 reg     [21:0]  tb_cnt      ;
22 reg             key_in      ;
23 reg     [1:0]   cnt_key     ;
24 reg     [3:0]   key         ;
25
26 //defparam  define
27 defparam    top_dds_inst.key_control_inst.CNT_MAX = 24;
28
29 //****************************************************************//
30 //************************** Main Code ***************************//
31 //****************************************************************//
32 //sys_rst_n,sys_clk,key
```

```
33 initial
34     begin
35         sys_clk     =   1'b0;
36         sys_rst_n  <=   1'b0;
37         key <= 4'b0000;
38         #200;
39         sys_rst_n   <=   1'b1;
40     end
41
42 always #10 sys_clk = ~sys_clk;
43
44 //tb_cnt:按键过程计数器，通过该计数器的计数时间来模拟按键的抖动过程
45 always@(posedge sys_clk or negedge sys_rst_n)
46     if(sys_rst_n == 1'b0)
47         tb_cnt <= 22'b0;
48     else    if(tb_cnt == CNT_60MS)
49         tb_cnt <= 22'b0;
50     else
51         tb_cnt <= tb_cnt + 1'b1;
52
53 //key_in:产生输入随机数，模拟按键的输入情况
54 always@(posedge sys_clk or negedge sys_rst_n)
55     if(sys_rst_n == 1'b0)
56         key_in <= 1'b1;
57     else    if((tb_cnt >= CNT_1MS && tb_cnt <= CNT_11MS)
58                 || (tb_cnt >= CNT_41MS && tb_cnt <= CNT_51MS))
59         key_in <= {$random} % 2;
60     else    if(tb_cnt >= CNT_11MS && tb_cnt <= CNT_41MS)
61         key_in <= 1'b0;
62     else
63         key_in <= 1'b1;
64
65 always@(posedge sys_clk or negedge sys_rst_n)
66     if(sys_rst_n == 1'b0)
67         cnt_key <=  2'd0;
68     else    if(tb_cnt == CNT_60MS)
69         cnt_key <=  cnt_key + 1'b1;
70     else
71         cnt_key <=  cnt_key;
72
73 always@(posedge sys_clk or negedge sys_rst_n)
74     if(sys_rst_n == 1'b0)
75         key     <=  4'b1111;
76     else
77         case(cnt_key)
78             0:      key <=  {3'b111,key_in};
79             1:      key <=  {2'b11,key_in,1'b1};
80             2:      key <=  {1'b1,key_in,2'b11};
81             3:      key <=  {key_in,3'b111};
82             default:key <=  4'b1111;
83         endcase
84
85 //*************************************************************//
86 //********************** Instantiation **********************//
87 //*************************************************************//
88 //------------- top_dds_inst -------------
89 top_dds top_dds_inst
90 (
91     .sys_clk    (sys_clk    ),
92     .sys_rst_n  (sys_rst_n  ),
93     .key        (key        ),
94
95     .dac_clk    (dac_clk    ),
96     .dac_data   (dac_data   )
97 );
98
```

```
99 endmodule
```

仿真波形分析

对顶层模块进行仿真，由图可知，输入与输出信号额能正常传入传出顶层模块，顶层模块通过仿真验证。

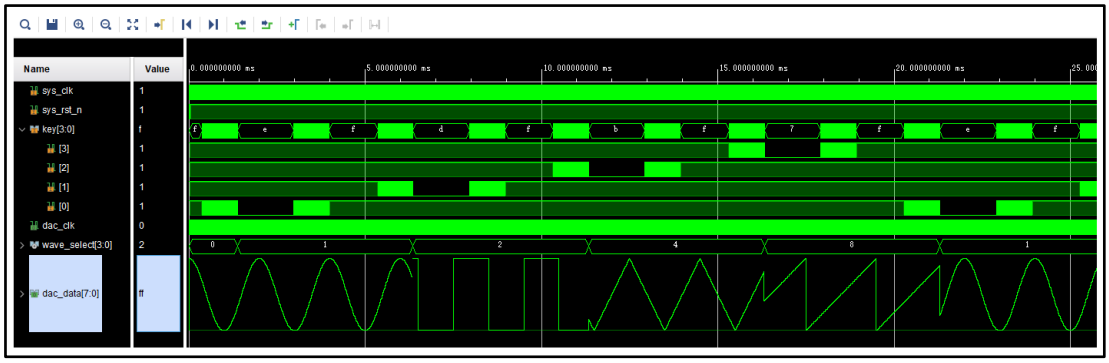

图 26-20 顶层模块仿真波形

26.3.4 上板验证

1. 引脚约束

仿真验证通过后，准备上板验证，上板验证之前先要进行引脚约束。工程中各输入输出信号与开发板引脚对应关系如表格 26-1 所示。

表格 26-1 引脚分配表

信号名	信号类型	对应引脚	备注
sys_clk	Input	W19	输入系统时钟
sys_rst_n	Input	N15	复位信号
key[3]	Input	AB18	按键 KEY4
key[2]	Input	AA18	按键 KEY3
key[1]	Input	W17	按键 KEY2
key[0]	Input	V17	按键 KEY1
dac_clk	Input	U22	输出至 DAC_CLK
dac_data[7]	Output	N19	输出至 DAC_D7
dac_data[6]	Output	N18	输出至 DAC_D6
dac_data[5]	Output	L18	输出至 DAC_D5
dac_data[4]	Output	M18	输出至 DAC_D4
dac_data[3]	Output	N14	输出至 DAC_D3
dac_data[2]	Output	N13	输出至 DAC_D2
dac_data[1]	Output	R19	输出至 DAC_D1
dac_data[0]	Output	P19	输出至 DAC_D0

引脚约束参考代码如代码清单 26-12

代码清单 26-12 引脚约束参考代码（top_dds.ucf）

```
1 #时钟复位
```

```
set_property PACKAGE_PIN W19 [get_ports sys_clk]
set_property PACKAGE_PIN N15 [get_ports sys_rst_n]
set_property IOSTANDARD LVCMOS33 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports sys_rst_n]

#按键信号
set_property PACKAGE_PIN V17 [get_ports {key[0]}]
set_property PACKAGE_PIN W17 [get_ports {key[1]}]
set_property PACKAGE_PIN AA18 [get_ports {key[2]}]
set_property PACKAGE_PIN AB18 [get_ports {key[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {key[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {key[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {key[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {key[0]}]

#DA 信号
set_property PACKAGE_PIN U22 [get_ports dac_clk]
set_property PACKAGE_PIN N19 [get_ports {dac_data[7]}]
set_property PACKAGE_PIN N18 [get_ports {dac_data[6]}]
set_property PACKAGE_PIN L18 [get_ports {dac_data[5]}]
set_property PACKAGE_PIN M18 [get_ports {dac_data[4]}]
set_property PACKAGE_PIN N14 [get_ports {dac_data[3]}]
set_property PACKAGE_PIN N13 [get_ports {dac_data[2]}]
set_property PACKAGE_PIN R19 [get_ports {dac_data[1]}]
set_property PACKAGE_PIN P19 [get_ports {dac_data[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports dac_clk]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dac_data[0]}]
```

2. 结果验证

如图 26-21 所示，开发板连接 12V 直流电源和 Xilinx 下载器 JTAG 端口；连接外载 AD/DA 板卡与开发板底板，连接 SMA 信号线。线路正确连接后，打开开关为板卡上电。

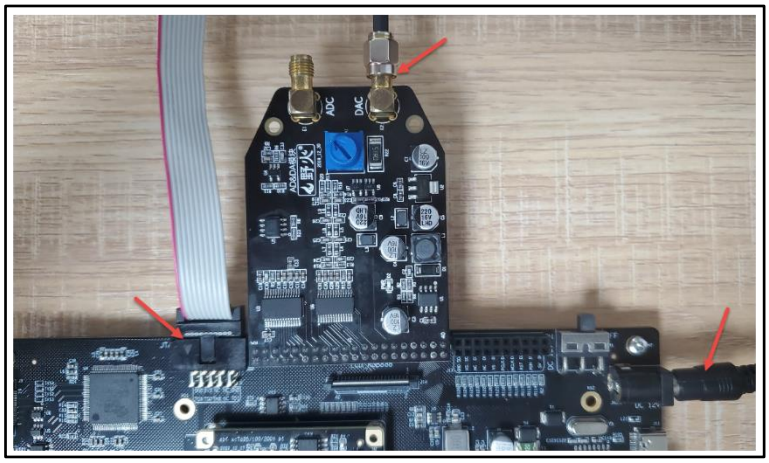

图 26-21 程序下载连线图

如图 26-22 所示，点击“Program”为开发板下载程序。

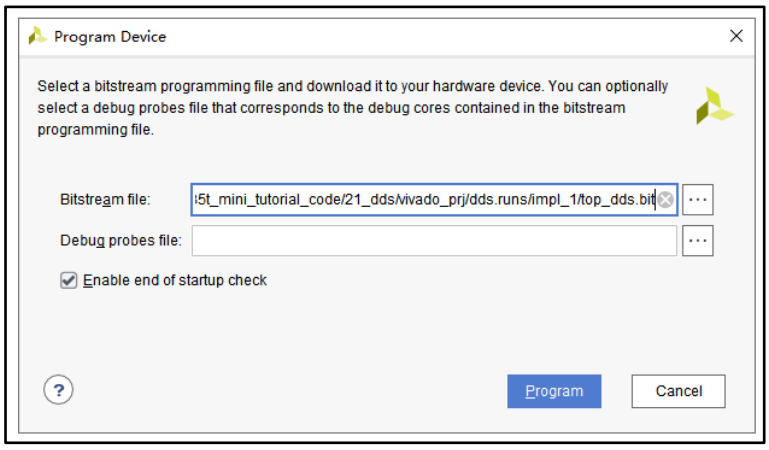

图 26-22 程序下载窗口

程序下载完成后，使用示波器对 AD/DA 板卡输出信号进行测量，如图 26-23 至图 26-26 所示，使用按键可进行输出波形切换，通过电位器可实现输出信号幅值调节。

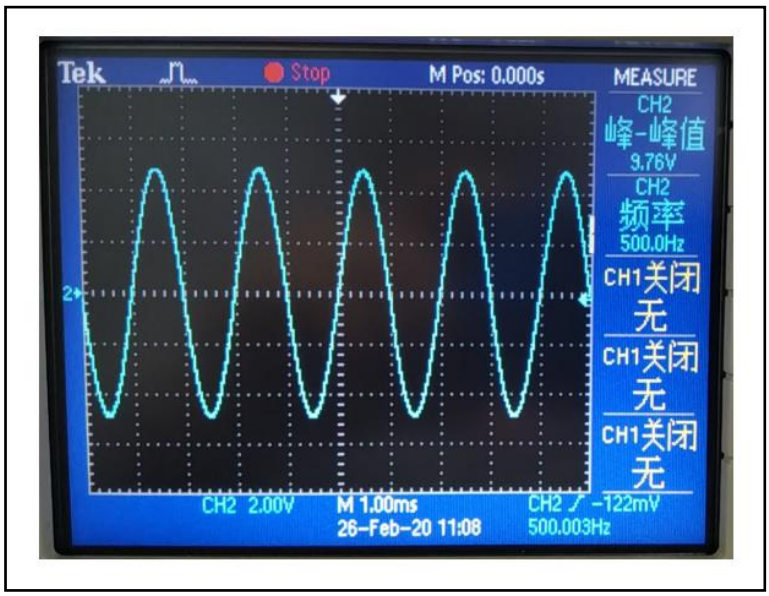

图 26-23 示波器测量图（一）

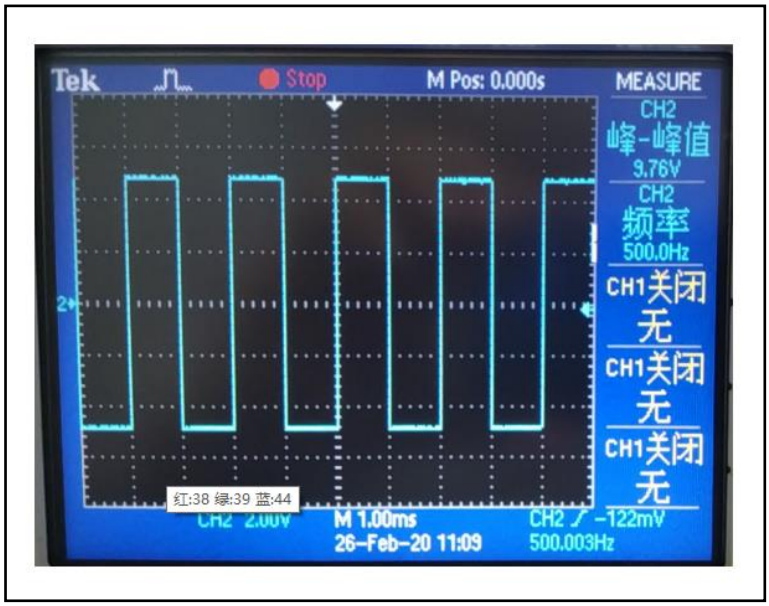

图 26-24 示波器测量图（二）

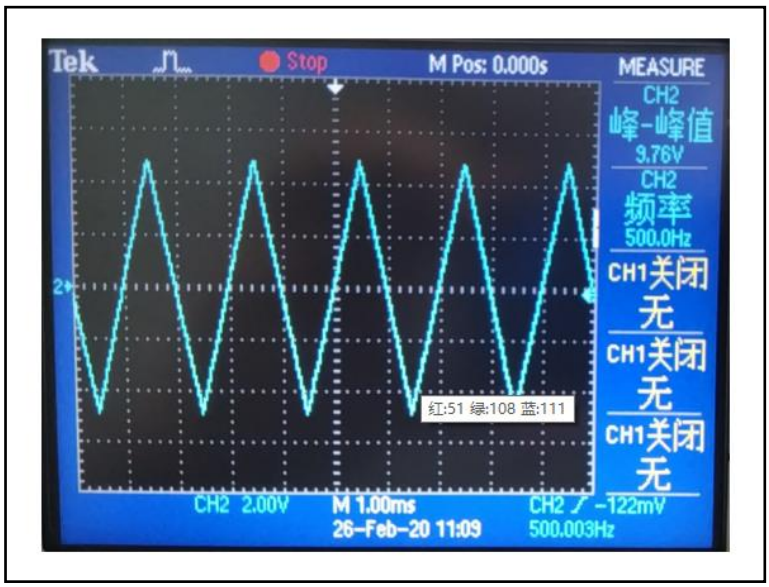

图 26-25 示波器测量图（三）

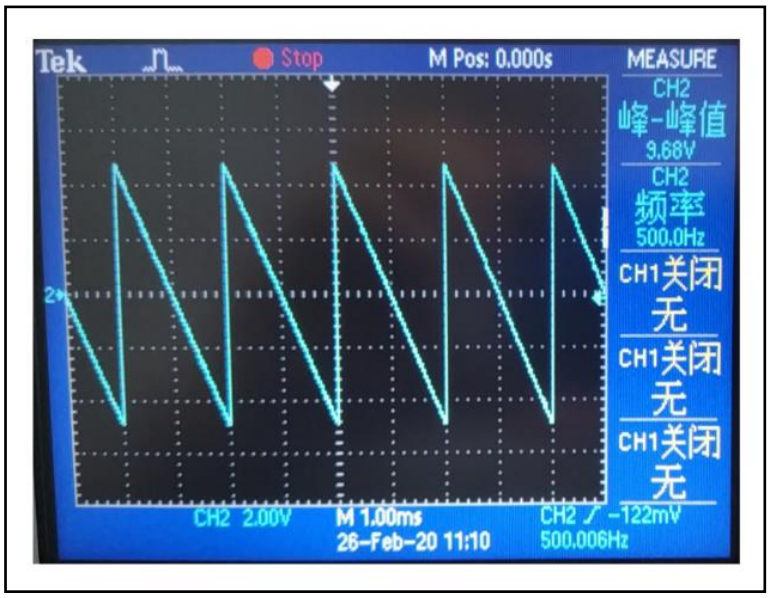

图 26-26 示波器测量图（四）

26.4 章末总结

在本章节中，我们实现了简易 DDS 信号发生器的设计与验证，并通过对简易 DDS 信号发生器的设计与验证这一实验工程讲解，为读者介绍了 DDS 信号发生器的相关内容和 DAC 芯片的相关知识，希望读者认真学习、切实掌握相关知识。

26.5 拓展训练

尝试按照公式计算并修改代码中的参数 FREQ_CTRL（频率输入字 K）和参数 PHASE_CTRL（相位输入字 P），实现输出不同频率、不同初相位的信号波形。