19 基于 AD9767 高速 DAC 的 DDS 信号发生器设计

章节导读

本节将介绍基于 ACM9767 模块的 DDS 数字信号发生器原理及设计方法，通过本节课程的学习，读者能够对基于高速 DA 转换器的数字信号发生器设计有更深入的理解。

19.1DDS 概述

DDS(Direct Digital Synthesizer)即数字合成器，是近年来发展起来的一种新的频率合成技术，其主要优点是相对带宽很大，频率转换时间极短（可小于 20 ns)，频率分辨率很高，全数字化结构便于集成，输出相位连续可调，且频率、相位和幅度均可实现程控。随着 FPGA 技术的不断发展，该技术得到愈加成熟的应用。借助 FPGA 平台开发的高性能的 DDS 发生器与基于 DDS 芯片设计的信号发生器相比，具有成本更低，操作更加灵活，而且还能根据要求在线更新配置等诸多优点。同时，整个系统开发流程更趋于软件化、自定义化，开发周期更短，调试过程更加简单。

正是由于其便捷的频率、相位以及幅度的数控优势，基于 FPGA 控制的 DDS 信号发生器有其广泛的应用前景。

19.2ACM9767 模块概述

提到基于 FPGA 的 ACM9767 模块波形信号发生器设计，就不得不先对 ACM9767 模块进行简单的介绍。

ACM9767 模块使用的是 ADI 公司 AD9767 型 DAC 芯片。该芯片支持 I、Q 输出模式（该模式常用于数字通信领域）。 输出形式为差分电流输出，输出电流满量程范围为可设置为 2~20mA。芯片本身自带 1.2V 的参考电压，无需外部提供参考源。

继承了 AD9767 芯片的优异性能，ACM9767 模块是一款高性能高速双通道 DAC 模块。 模块具有单电源 5V 供电输入，双通道数字转模拟信号输出，每个通道数据分辨率为 14 位， 输出电压范围为+-5V， 且转换速率高达 125Msps，非常适合诸如信号发生器、数字调制通信系统的开发等应用。

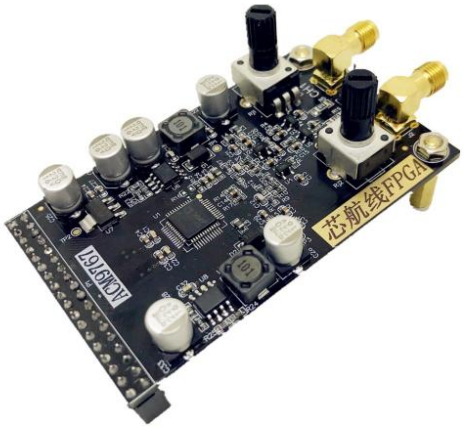


图 19-1 ACM9767 模块样图

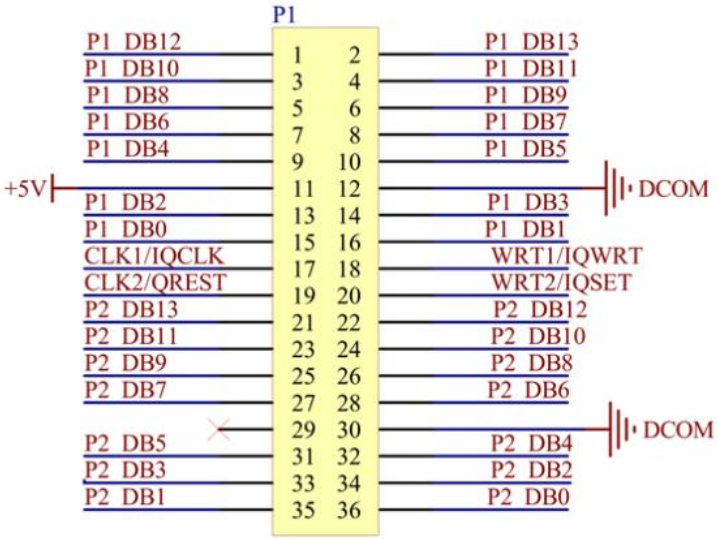


图 19-2 ACM9767 模块插接管脚信号定义

模块对用户提供一个 36 针的排母接口，可直接插接到市面上各种常见的 FPGA 开发板（Terasic 公司 DE2、DE1、 DE0，芯路恒科技（小梅哥 FPGA）所有的 FPGA 开发板，包含 ACX720、AC620、 AC6103、 AC601、AC609 等）上进行使用，而无需使用任何转接线转接。

本次系统设计将在小梅哥团队出品的 AC6103 开发板上进行。在使用时，将 ACM9767 模块插接到 AC6103 开发板的通用 GPIO 接口上，即可实现 FPGA 和 ACM9767 模块的数字接口互通。由于在 ACM9767 模块模拟量输出峰值范围内，输出的模拟量电压值可以由输入的数字量信号完全控制，这样，就可以通过 FPGA 完全控制数字量的输出值，实现满足预定指标的模拟信号波形输出。

基于此，借助 ACM9767 模块和 AC6103 开发板实现：输出波形可以是不同频率、幅度的正弦波、三角波、方波信号，同时用户可以根据实际需求通过按

键对波形的频率和相位值进行调整切换这一设计功能，是完全可行的。

接下来，我们将从典型的 DDS 信号发生器信号发生流程图，着手以上设计功能的实现。

19.3系统原理及整体设计

19.3.1DDS 信号发生器的系统设计原理

介绍完 DDS 信号发生的应用背景和 ACM9767 的模块硬件性能，接下来我们来共同分析和探讨基于 FPGA 的 DDS 信号发生器的实现过程。

DDS 是如何实现控制发生信号的呢？如果想生成一个质量较好的 DDS 发生信号，无非是对波形、频率、相位这几个关键要素进行控制和搭配组合。下面是其中一种常用的 DDS 控制信号发生器的实现方案原理图。

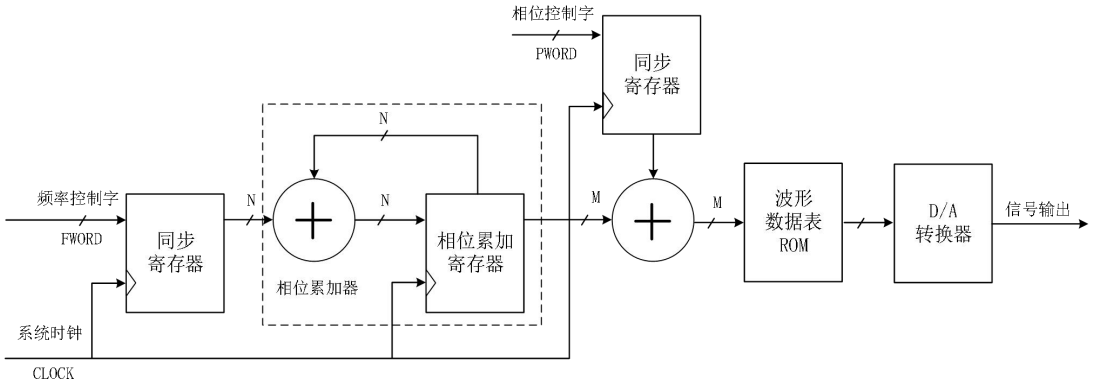


图 19-3 DDS 控制信号发生原理图

由图可以看出，DDS 主要由相位累加器、相位调制器、波形数据表以及 D/A 转换器构成。其中相位累加器由 N 位加法器与 N 位寄存器构成。每个时钟周期的时钟上升沿，加法器就将频率控制字与累加寄存器输出的相位数据相加，相加的结果又反馈至累加寄存器的数据输入端，以使加法器在下一个时钟脉冲的作用下继续与频率控制字相加。这样，相位累加器在时钟作用下，不断对频率控制字进行线性相位累加。即在每一个时钟脉冲输入时，相位累加器便把频率控制字累加一次。

相位累加器输出的数据就是合成信号的相位。相位累加器的溢出频率，就是 DDS 输出的信号频率，相位累加器输出的数据，作为波形存储器的相位采样地址，这样就可以把存储在波形存储器里的波形采样值经查表找出，完成相位到幅度的转换。

波形存储器的输出数据送到 D/A 转换器，由 D/A 转换器将数字信号转换成模拟信号输出。

DDS 信号流程示意图如下图 19-4 所示：

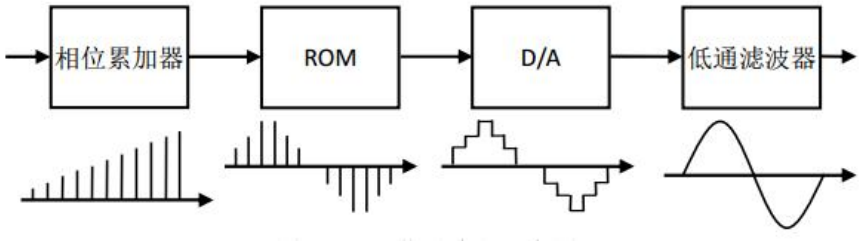


图 19-4 DDS 信号流程示意图

这里相位累加器位数为 N 位（N 的取值范围实际应用中一般为 24~32），相当于把正弦信号在相位上的精度定义为 N 位，所以其分辨率为$1/2^N$。

若 DDS 的时钟频率为F_{clk}，频率控制字 fword 为 1，则输出频率为$F_{out}=\frac{F_{clk}}{2^N}$，这个频率相当于“基频”。若 fword 为 B，则输出频率为$F_{out}=B\times\frac{F_{clk}}{2^N}$。

因此理论上由以上三个参数就可以得出任意的f_o输出频率。且可得出频率分辨率由时钟频率和累加器的位数决定的结论。当参考时钟频率越高，累加器位数越高，输出频率分辨率就越高。

从上式分析可得，当系统输入时钟频率F_{clk}不变时，输出信号频率由频率控制字 B 所决定，由上式可得：$B=2^N\times\frac{F_{out}}{F_{clk}}$。其中 B 为频率字且只能取整数。为了合理控制 ROM 的容量，此处选取 ROM 查询的地址时，可以采用截断式，即只取 32 位累加器的高 M 位。这里相位寄存器输出的位数一般取 10~16 位。

19.3.2DDS 信号发生器原理讲解举例

以上通过理论计算加数据变换的形式对 DDS 原理进行了较为严谨的公式推导解释，但是如果从采样点选取的角度分析，DDS 究竟是怎么实现频率和相位的控制的呢？以下通过一个简化的实例来描述 DDS 实现频率和相位控制的过程。

如下图 19-5 为一个完整周期的正弦信号的波形，总共有 33 个采样点，其中第 1 点和第 33 点的值相同，第 33 点为下一个周期的起始点，因此，实际一个周期为 32 个采样点（1~32）。

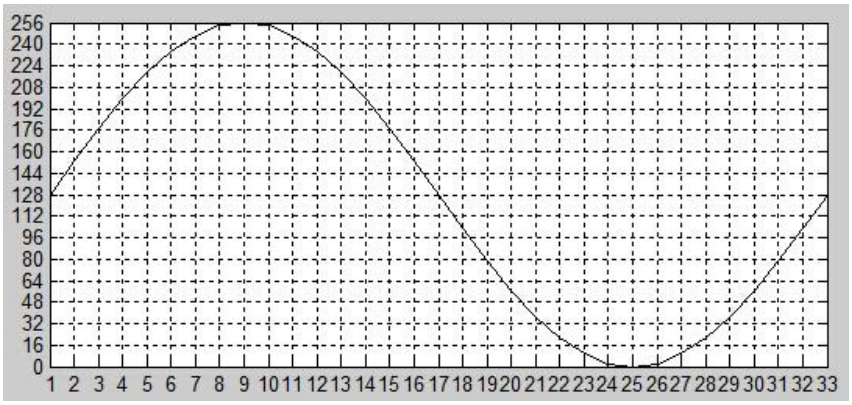


图 19-5 完整周期的正弦信号波形示例

例如：当使用 FPGA 控制 DAC 输出一个周期的正弦信号时，每 1ms 输出一个数值。如果每个点都输出，则总共输出这一个完整的周期信号需要输出 32 个点，因此输出一个完整的信号需要 32ms，可知输出信号的频率为 1000/32Hz。

如果需要用这一组数据来输出一个 2*（1000/32）Hz 的正弦信号，因为输出信号频率为 2*（1000/32）Hz，那么输出一个完整的周期的正弦波所需要的时间为 32/2，即 16ms。为了保证输出信号的周期为 16ms，我们需要对我们的输出策略进行更改，上面输出周期为 32ms 的信号时，我们采用的为逐点输出的方式，以 32 个点来输出一个完整的正弦信号，而我们 FPGA 控制 DAC 输出信号的频率固定为 1ms。因此，我们要输出周期为 16ms 的信号，只能输出 16 个点来表示一个完整的周期。我们就选择以每隔一个点输出一个数据的方式来输出即可。我们可以选择输出（1、3、5、7……29、31）这些点，因为采用这些点，我们还是能够组成一个完整周期的正弦信号，而输出时间缩短为一半，即频率提高了一倍。最终结果如下图 19-6 所示：

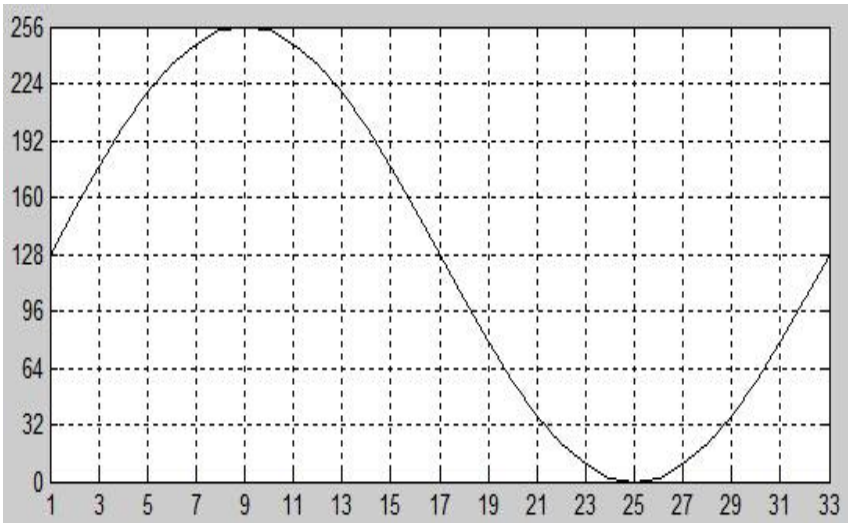


图 19-6 频率提高 1 倍后的正弦输出信号

如果需要使用该组波形数据输出频率为(1/2)*（1000/32）Hz 的信号，即周期为 64ms，则只需要以此组数据为基础，每 2ms 输出一个数据即可，例如第 1ms 和第 2ms 输出第一个点，第 3ms 和第 4ms 输出第二个点，以此类推，第 63ms 和第 64ms 输出第 32 个点，即可实现周期加倍，即频率减半的效果。

概括来说，一个周期的离散点数值表格一定，如果想提高输出频率，则采用跳过若干点位（1,3,5.....）的方法，重构原波形；如果想降低输出频率，则采用拉长每一个离散点的输出时间(1,1,2,2,3,3.....)，来完成输出波形不变，而扩大时钟周期的输出效果。时钟周期扩大，则频率降低。

对于相位的调整，则更加简单。只需要在每个取样点的序号上加上一个偏移量，便可实现相位的控制。例如，上述一个周期 32 个点均匀的将 360 度等分，上面默认的是第 1ms 时输出第一个点的数据，假如我们现在在第 1ms 时从第 9 个点开始输出，则将相位左移了 90 度，这就是控制相位的原理。

在本次设计中，对于一个周期的离散点数值表格，我们实现 DDS 输出时，将横坐标上的数据作为 ROM 的地址，纵坐标上的数据作为 ROM 的输出，那么指定不同的地址就可实现对应值的输出。而我们 DDS 输出控制频率和相位，归结到底就是控制 ROM 的读出地址。

在理解了 DDS 的实现原理之后，接下来便可以开始本次系统的设计实现了。

19.3.3信号发生控制方案设计框图

介绍完 DDS 信号发生的原理，并进行了举例后，接下来将进行 DDS 信号发生器的整体设计。

根据以上原理分析，基于 AC6103 开发板，我们可以作如下架构设计

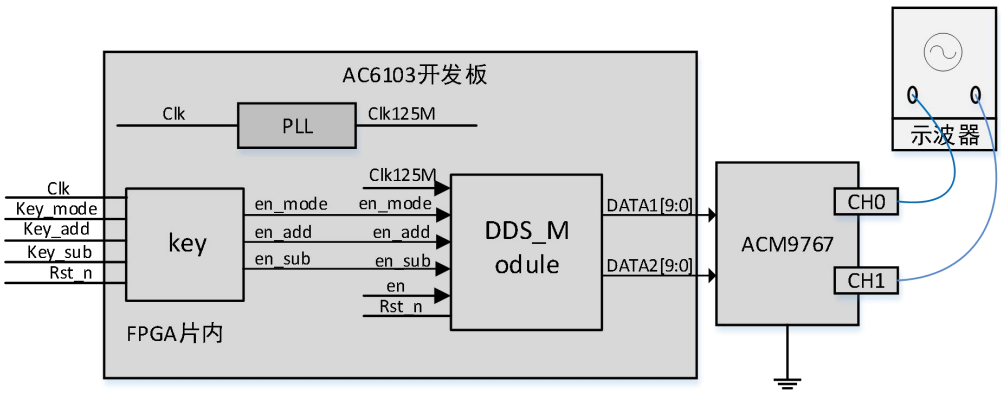


图 19-7 基于 AC6103 开发板的 ACM9767 模块系统框图

为了验证频率和相位可以在一段区间内实现变化这一特性，我们可以通过

开发板上的按键按下切换档位的方式，设定几个可以跳变切换的典型值以验证频率相位的实时变化效果。但是，由于 AC6103 开发板上的按键有限，所以本次实验，我们设计时可以先按如下规律实现设计：

1. 程序上电后给出默认频率和相位初始值。
2. 每次按下按键 S0，切换一种模式。正常情况下按一下，进入 ACM9767 模块的 CH1 通道波形选择，按第二下，进入 ACM9767 模块的 CH2 通道波形选择，按第三下，进入波形频率设置模式，按第四下，恢复正常模式。
3. 在正常模式下按下按键 S1 无作用，在波形模式下，切换波形，在频率设置模式下，切换频率（增加）。
4. 在正常模式下按下按键 S2 无作用，在波形模式下，切换波形，在频率设置模式下，切换频率（减小）。

开发板上的按键用途说明如下表 19-1 所示。

表 19-1 按键用途设计说明表

按键名称	按键用途
S0	切换模式
S1	增加
S2	减小

在曾经的学习内容中，我们也介绍了按键消抖的相关理论和使用场景。在本工程中，由于同样涉及到人工按下按键属于随机触发的外部信号的情况，我们同样需要加入按键消抖的相关处理模块来解决按键抖动的问题。

明确了频率和相位控制字切换的设计方案后，还需要明确波形信号的存储和读取策略。为了便于波形数据取用的统一管理，对于 3 种不同类型的波形数据存储，我们可以为每个信号发生通道开辟 3 个 ROM 空间。这三个 ROM 空间分别存储正弦波、方波和三角波的离散信号值作信号发生时查表使用。

在设计之初，设计人员也许有过这样的思考：我能否只开辟一块 ROM 空间，然后输出哪种波形，就装载哪种波形的 ROM 表？这样做可以节省 FPGA 片上的 ROM 空间资源，让更多 ROM 空间资源预留给其他需要进行缓存的设计单元。

但是，这样做会产生如下问题：熟悉 FPGA 的 ROM IP 核的用户都知道，切换 ROM 表的装载文件，即 mif 文件，不是一件容易的事情，每切换一次 mif 文件，都需要对 ROM IP 核进行 mif 文件的重新装载，并重新编译 IP 核和工程。

这样看来，要想让已经完成编译并下载到 FPGA 开发板内的工程进行实时的发生信号波形切换，就变成了难以完成的任务。所以，从波形实时切换和选择的角度，只有并行开辟三块 ROM 空间并加载 mif 文件，然后同时读取波形数据，并在最终输出端进行输出波形数据选择，才能实现信号的实时切换功能。

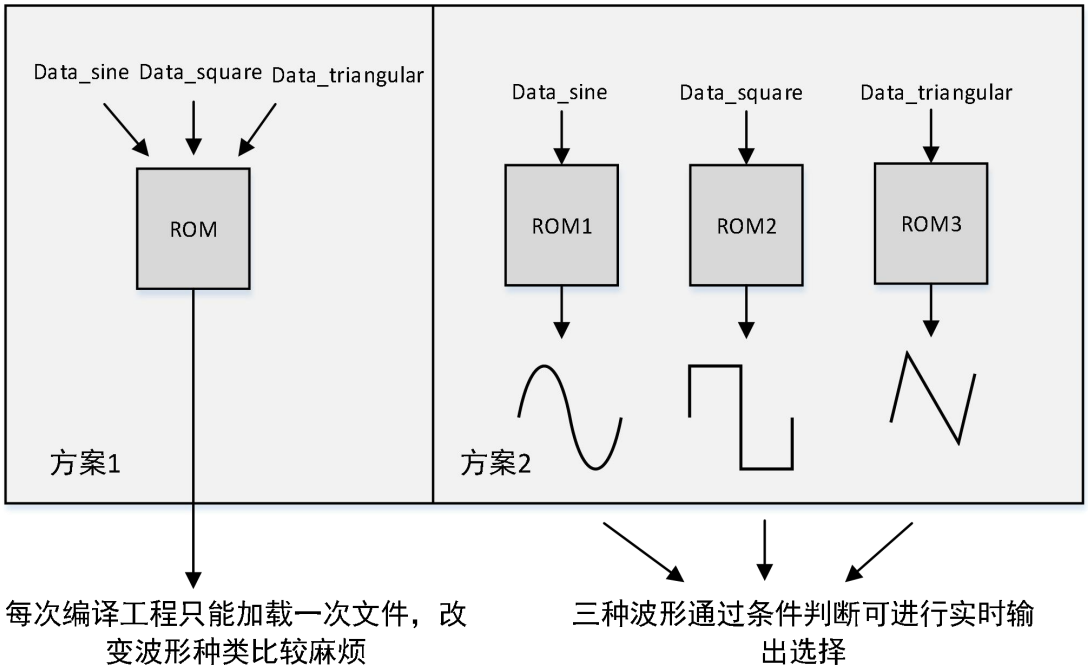


图 19-8 方案 1 和方案 2 模块架构对比分析

通过以上分析比较发现，使用方案 2 比使用方案 1 虽然占用更多的 ROM 资源，但输出波形的实时响应性能更优秀，所以在 ROM 资源充沛的情况下，优先选用方案 2。

19.4系统各模块设计

在有了系统总体框架思路后，接下来即可着手对模块各个击破。经过上述分析，我们在本次设计中需要利用按键消抖模块解决按键按下抖动的问题，同时，我们需要配置锁相环模块、信号离散值 ROM 表、按键控制模块以及对 DDS 驱动模块进行设计。由于按键消抖模块已经有专门的章节进行讲解，在这里，受篇幅所限，我们也不再赘述，而将讲解重点放在如何设计 ACM9767 驱动模块及几个重要 IP 核的配置之中。

19.4.1锁相环模块

由于 AD9767 的工作时钟频率为 125MHz，所以这里我们需要利用 FPGA

内部的锁相环为 ACM9767 模块以及其 FPGA 配套的驱动模块提供相同的工作频率。这样，在数据交互时，能确保时钟域相同。

锁相环的配置方法我们也讲过多次，我们也就不再进一步介绍。这里，我们给出配置界面供参考即可。

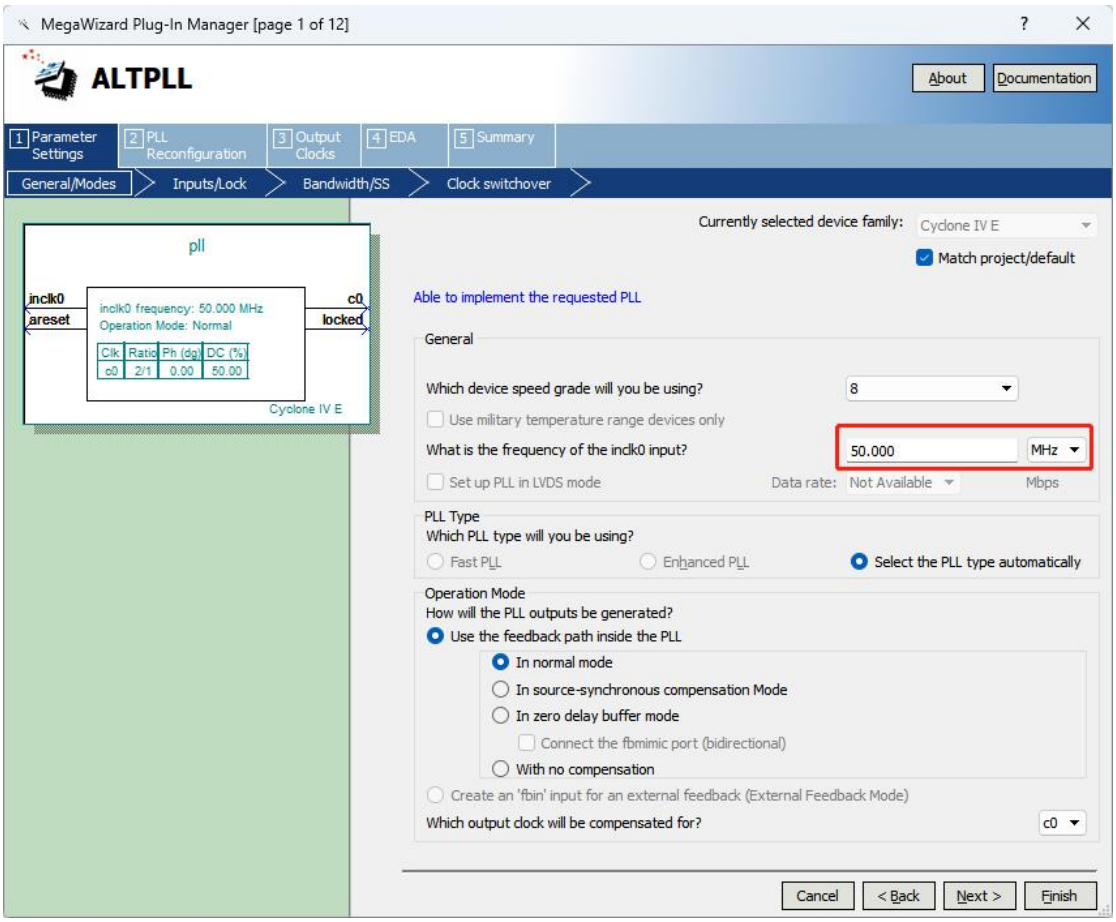


图 19-9 锁相环配置界面 1

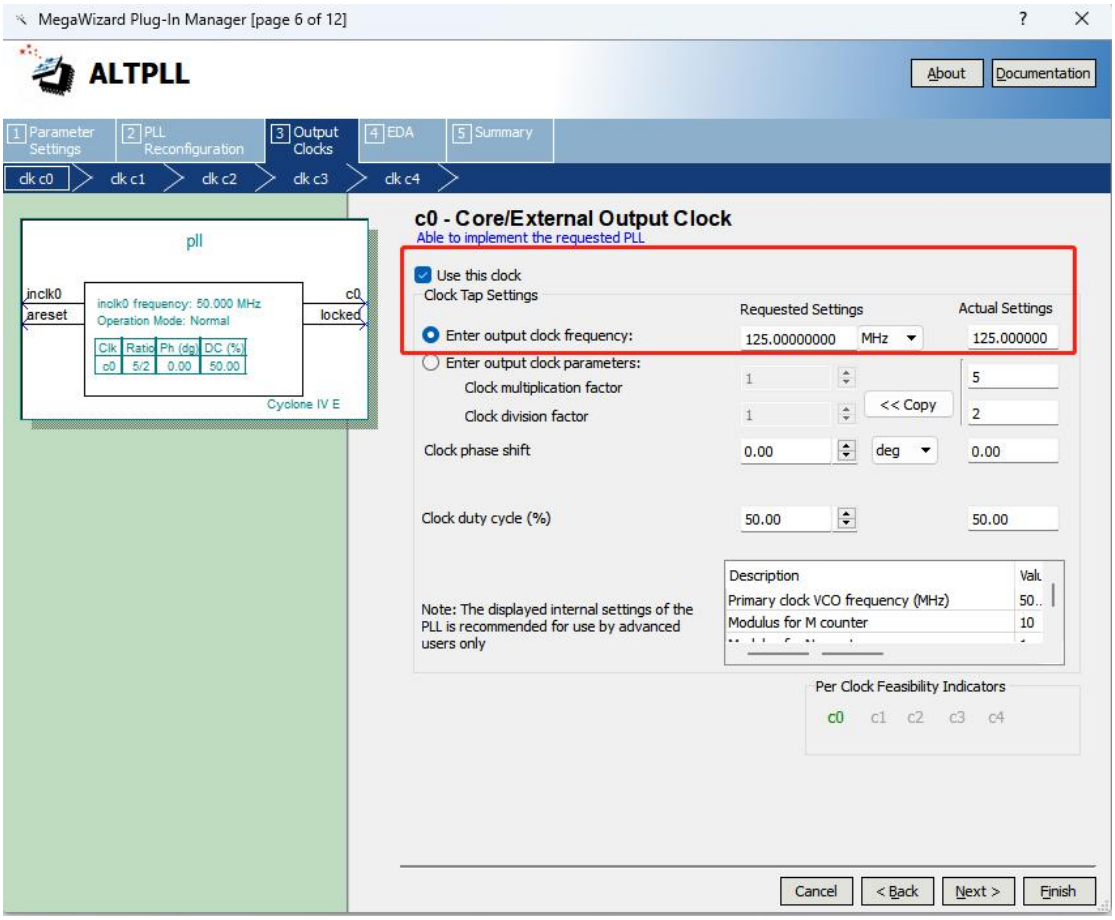


图 19-10 锁相环配置界面 2

完成配置后，即可利用锁相环将 50M 主时钟频率，通过分频倍频而获得我们需要的 125MHz 时钟。

19.4.2波形数据存储单元

为了能产生相应的波形，我们需要借助 ROM 来分别存储正弦波、方波、三角波的波形数据。其中单端口的 ROM 主要配置数据如下图 19-11 所示，其初始化文件选为已经生成的相应波形的 mif 文件。此处 mif 的位宽为 10，深度为 4096。

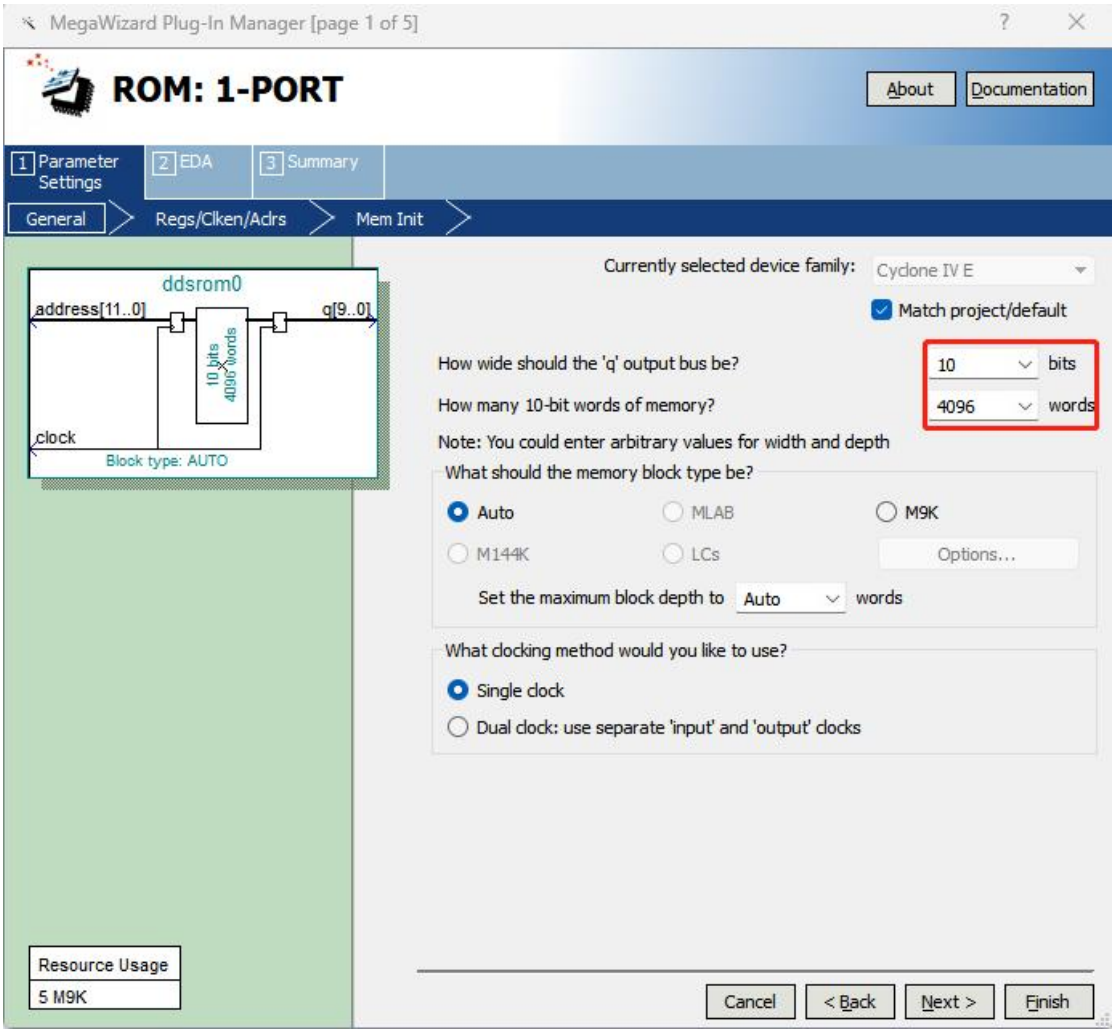


图 19-11 波形数据存储 ROM 配置表

关于 mif 文件的生成可以参看“IP 核使用之 ROM”一节的内容。同时，由于 AD 芯片的数字位宽为 10 位，所以 maxi 中数据改为 1023（即 2^10 - 1）可以确保 FPGA 输出的二进制值，都能反映到输出模拟量实时输出值的变化之中，其余设置如下图所示，然后剩下的按照“IP 核使用之 ROM”一节的内容进行修改。

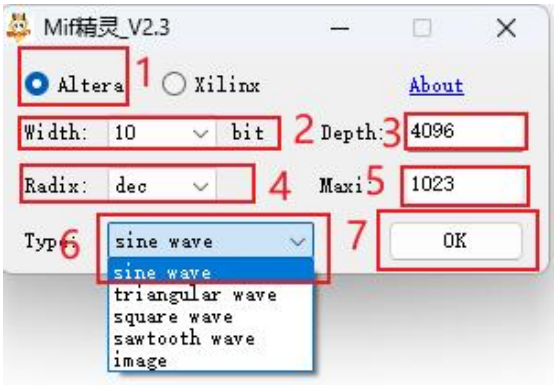


图 19-12 使用 Mif 精灵生成波形信号参考图示

19.4.3DDS_Module 模块设计

在本设计中参考时钟F_{clk}频率为 125MHz，相位累加器位数 N 取 32 位，频率控制字位数 M 取 12 位。经过以上的分析，可以得出 DDS 模块的端口示意图如下图 19-13 所示。

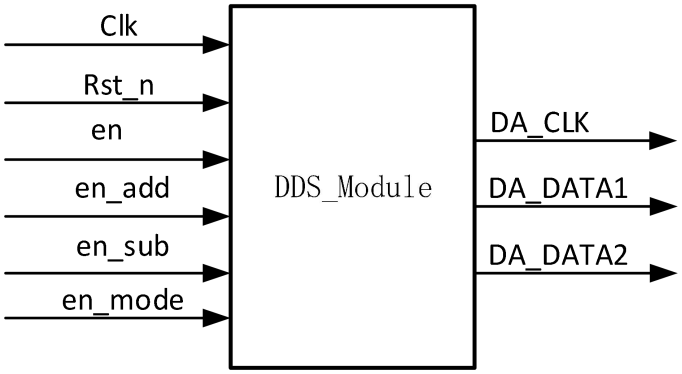


图 19-13 DDS_Molude 模块端口

其中，每个端口的功能描述如下所示。

表 19-2 DDS_Molude 模块端口列表

端口名称	I/O	功能描述
Clk	I	为本模块的工作时钟，频率为锁相环输出的 125MHz
Rst_n	I	控制器复位，低电平复位
en	I	使能信号
en_add	I	按键增加使能信号
en_sub	I	按键减小使能信号
en_mode	I	按键切换模式使能信号
DA_CLK	O	DAC 的时钟
DA_DATA1	O	DAC 输出的数据 1
DA_DATA2	O	DAC 输出的数据 2

为了研究简便，我们以双通道信号发生器的其中一个通道作为研究对象，而另一个通道与之完全相同。实际在 FPGA 驱动 ACM9767 工作过程中，FPGA 设计上只需重复例化驱动模块一次，在管脚绑定时只需按双通道对应的数字信号驱动管脚驱动外部 ACM9767 模块即可。

接下来，开始模块的代码设计讲解。

为了便于对频率和相位控制字的后期处理，在驱动模块之中必须先对频率和相位值进行缓存。其代码如下：

```
//频率控制字同步寄存器
reg [31:0]Fword_r;
```

```
always@(posedge Clk)
Fword_r <= Fword;

//相位控制字同步寄存器
reg [11:0]Pword_r;
always@(posedge Clk)
Pword_r <= Pword;
```

实现波形数据的还原，其实质就是按频率控制字每个时钟周期进行累加，进而决定下一个时钟周期应该便宜多少个周期切分的单元，从而作为下一次还原的数据。

```
//相位累加器
reg [31:0]Freq_ACC;
always@(posedge Clk or negedge Reset_n)
if(!Reset_n)
    Freq_ACC <= 0;
else
    Freq_ACC <= Fword_r + Freq_ACC;
```

需要注意的是，在该模块设计中我们直接截取 32 位累加器结果中的高 12 位作为 ROM 的查询地址，这样，查询地址也是 4096 个点，和 ROM 表保存的单周期的离散点存储深度是一致的。这样产生的误差虽然会对频谱纯度有影响，但是对波形的精度的影响是可以忽略的。

```
//波形数据表地址
reg [11:0]Rom_Addr;
always@(posedge Clk)
        Rom_Addr <= Freq_ACC[31:20] + Pword_r;
```

本工程设计共创建了 3 种典型的发生信号模型，如果希望输出哪一种波形，则进行对应的模式选择即可。从这个角度来看，每个通道的 3 块 ROM 空间，实际上都在同时读取数据和输出数据，至于最终呈现在 FPGA 输出端的到底是何种波形的数据，则由模式选择信号 en_mode 决定。

```
always @ (posedge clk or negedge rst_n)
if (!rst_n)
    mode <= 2'd0;
else if (en_mode)
    mode <= mode + 1'd1;
else
    mode <= mode;

/*---------------通道 0------------------*/
always @ (posedge clk or negedge rst_n)
if (!rst_n)
   wave_a <= 2'd0;          //正弦波
else if (mode == 2'd1)
```

```
    if (en_sub)
        if (mode > 2'd0)
            wave_a <= wave_a - 1'd1;
        else
            wave_a <= 2'd2;
    else if (en_add)
        if (mode < 2'd2)
            wave_a <= wave_a + 1'd1;
        else
            wave_a <= 2'd0;
else
    wave_a <= wave_a;

/*---------------通道 1------------------*/
always @ (posedge clk or negedge rst_n)
if (!rst_n)
    wave_b <= 2'd1;             //矩形波
else if (mode == 2'd2)
    if (en_sub)
        if (mode > 2'd0)
            wave_b <= wave_b - 1'd1;
        else
            wave_b <= 2'd2;
    else if (en_add)
        if (mode < 2'd2)
            wave_b <= wave_b + 1'd1;
        else
            wave_b <= 2'd0;
else
    wave_b <= wave_b;
```

通过上述代码，我们就可以通过三个按键选择我们所需要的波形，记住，这里用了三个按键，那免不了使用按键消抖，这里我就不进行介绍了。

接下来就是调频，也是通过模式选择信号 en_mode 来实现，具体代码实现如下：

```
/*---------------按键调频------------------*/
always @ (posedge clk or negedge rst_n)
if (!rst_n)
    freq <= 3'd0;
else if (mode == 2'd3)
    if (en_sub)
        freq <= freq - 1'd1;
    else if(en_add)
        freq <= freq + 1'd1;
else
    freq <= freq;
```

```
always @ (posedge clk or negedge rst_n)
if (!rst_n)
   Fword <= 32'd34360;              //默认 1khz
else
   case(freq)
      0:Fword <= 32'd34360;         //1k
      1:Fword <= 32'd343597;        //10k
      2:Fword <= 32'd1717985;       //50k
      3:Fword <= 32'd3435974;       //100k
      4:Fword <= 32'd17179870;      //500k
      5:Fword <= 32'd34359738;      //1m
      6:Fword <= 32'd171798690;     //5m
      7:Fword <= 32'd343597384;     //10m
  endcase
```

这里我们给定几个固定的频率，分别为 1K、10K、50K、100K、500K、1M、5M、10M。通过按键按下按键 S0，进入波形频率设置模式，再通过按下按键 S1 和 S2 则可以改变波形输出的频率。

接下来就是控制 AD9767 双通道输出的数据，具体代码如下：

```
always @ (posedge clk or negedge rst_n)
if (!rst_n)
   begin
      DA_DATA1 <= wave_sin;
      DA_DATA2 <= wave_sin;
   end
else
   case ({wave_a,wave_b})
   4'b0000:begin DA_DATA1 <= wave_sin; DA_DATA2 <= wave_sin; end
   4'b0001:begin DA_DATA1 <= wave_sin; DA_DATA2 <= wave_square; end
   4'b0010:begin DA_DATA1 <= wave_sin; DA_DATA2 <= wave_triangle; end
   4'b0100:begin DA_DATA1 <= wave_square; DA_DATA2 <= wave_sin; end
   4'b0101:begin DA_DATA1 <= wave_square; DA_DATA2 <= wave_square; end
   4'b0110:begin DA_DATA1 <= wave_square; DA_DATA2 <= wave_triangle; end
   4'b1000:begin DA_DATA1 <= wave_triangle; DA_DATA2 <= wave_sin; end
   4'b1001:begin DA_DATA1 <= wave_triangle; DA_DATA2 <= wave_square; end
   4'b1010:begin DA_DATA1 <= wave_triangle; DA_DATA2 <= wave_triangle;end
   default:begin DA_DATA1 <= DA_DATA1; DA_DATA2 <= DA_DATA2; end
   endcase
```

接下来就是例化三个 ROM IP 核。

```
/*----------例化查找表 ROM-------*/
ddsrom0 ddsrom_sin(
   .address(Rom_Addr),
   .clock(clk),
   .q(wave_sin)
);
```

```
ddsrom1 ddsrom_square(
    .address(Rom_Addr),
    .clock(clk),
    .q(wave_square)
);

ddsrom2 ddsrom_triangle(
    .address(Rom_Addr),
    .clock(clk),
    .q(wave_triangle)
);
```

至此 DDS_Module 模块设计完成。

19.4.4工程顶层模块设计

在工程顶层，我们实现了各子模块的例化，具体代码如下：

```
module AC6103_ACM9767_DDS(
    ref_clk,
    key_mode,
    key_add,
    key_sub,
    daca_clk,
    dacb_clk,
    daca_wrt,
    dacb_wrt,
    dac_data1,
    dac_data2
);

    input ref_clk;
    input key_mode;
    input key_add;
    input key_sub;

    output daca_clk;
    output dacb_clk;
    output daca_wrt;
    output dacb_wrt;

    output [13:0] dac_data1, dac_data2;

    wire clk;
    wire rst_n;
    wire en_beep;
    wire en_mode;
```

```
    wire en_sub;
    wire en_add;

    wire [31:0]Fword0,Fword1;//频率控制字
    wire [11:0]Pword0,Pword1;//相位控制字

    wire [9:0]DA_Data1,DA_Data2;//DDS 输出的波形数据

    assign dac_data1 = {DA_Data1,4'd0};
    assign dac_data2 = {DA_Data2,4'd0};

    assign daca_clk = clk;
    assign dacb_clk = clk;

    assign daca_wrt = clk;
    assign dacb_wrt = clk;

    pll PLL(
        .inclk0(ref_clk),
        .c0(clk),
        .locked(rst_n)
    );

    key KEY(
        .clk(clk),
        .rst_n(rst_n),
        .key_mode(key_mode),
        .key_add(key_add),
        .key_sub(key_sub),
        .en_mode(en_mode),
        .en_add(en_add),
        .en_sub(en_sub)
    );

    DDS_Module DDS_CHANEL_AB(
        .clk(clk),
        .rst_n(rst_n),
        .en(1'b1),
        .en_mode(en_mode),
        .en_sub(en_sub),
        .en_add(en_add),
     //.DA_Clk(),
        .DA_DATA1(DA_Data1),
        .DA_DATA2(DA_Data2)
    );

endmodule
```

这里，我们的顶层模块也设计完成。

19.5激励创建及仿真测试

19.5.1激励信号设计

仿真 tb 文件可以作如下设计：

```
`timescale 1ns / 1ps

module AC6103_ACM9767_DDS_tb;

  reg  ref_clk;
  reg  key_mode;
  reg  key_add;
  reg  key_sub;

  wire daca_clk;
  wire dacb_clk;
  wire daca_wrt;
  wire dacb_wrt;
  wire  [13:0]dac_data1;
  wire  [13:0]dac_data2;

  AC6103_ACM9767_DDS   AC6103_ACM9767_DDS(
      .ref_clk(ref_clk),
      .key_mode(key_mode),
      .key_add(key_add),
      .key_sub(key_sub),
      .daca_clk(daca_clk),
      .dacb_clk(dacb_clk),
      .daca_wrt(daca_wrt),
      .dacb_wrt(dacb_wrt),
      .dac_data1(dac_data1),
      .dac_data2(dac_data2)
  );
  initial ref_clk = 1;
  always #10 ref_clk= ~ref_clk;

  initial begin
      key_mode = 0;
      key_add = 0;
      key_sub = 0;
      #10000000;
      $stop;
```

```
  end

endmodule
```

19.5.2仿真结果分析

根据以上仿真代码，可以得到如下图 19-16 仿真波形，这里我们对波形进行了分组，同时我们对关键信号进行监视，以有利于观察波形数据输出结果。

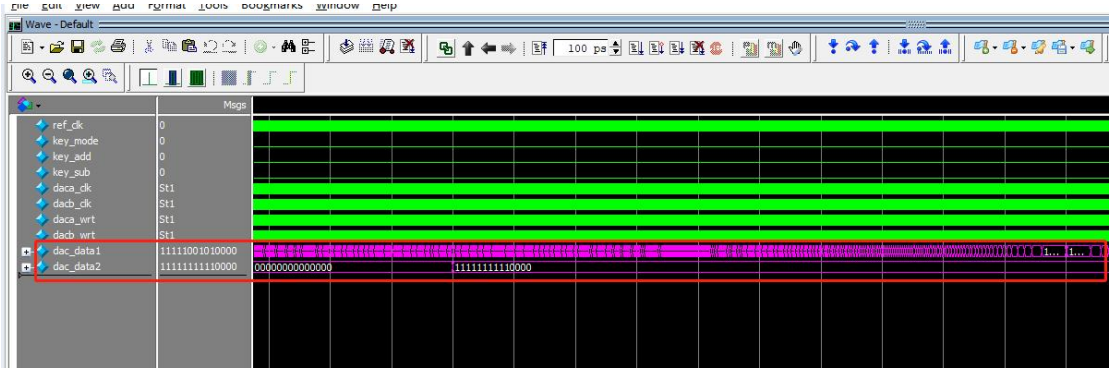
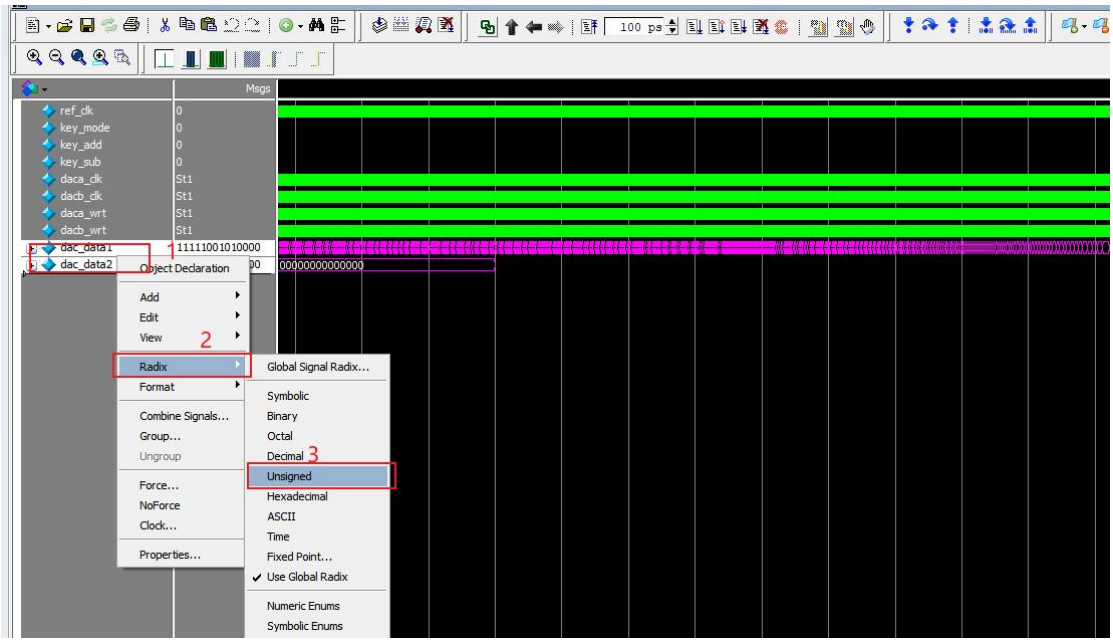


图 19-16 仿真波形总览

默认来说，软件给出的输出结果都是数字量，而对本实验来说，则是让仿真软件输出模拟量波形会更加直观展现我们的设计意图。Modelsim 仿真工具具备将数字量转换成模拟量的能力。这里我们只需要鼠标右键点击信号名称，然后切换波形输出类型即可。



先将信号改成无符号型。

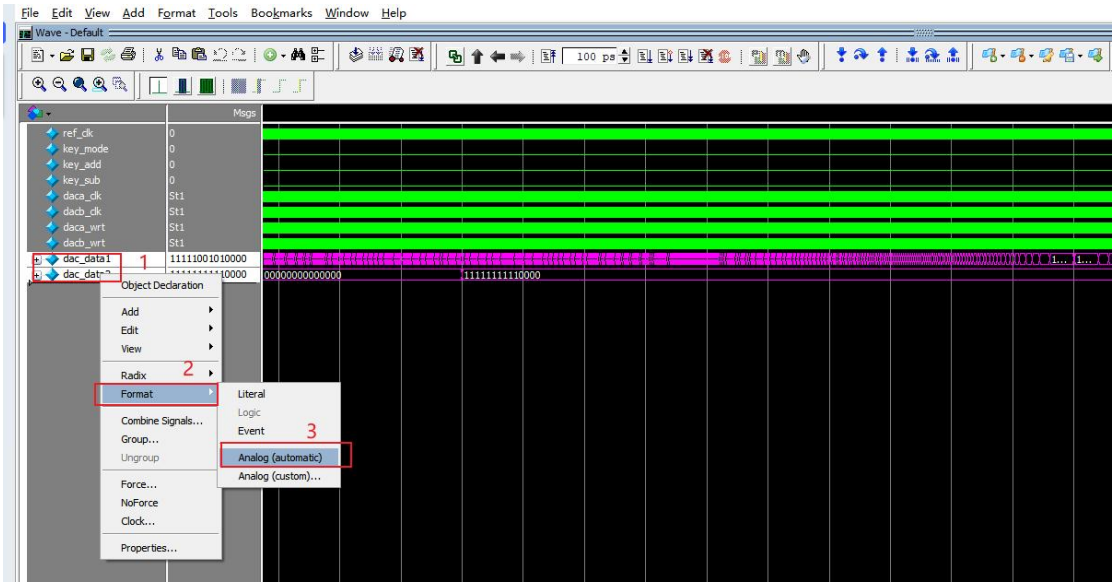


图 19-17 波形的数字量输出转换成模拟量输出

对于通道 1，经过转换后，可以看到，输出周期为 0.1ms 的正弦波，对于通道 2 输出 0.1ms 的方波，波形图如下图 19-18 所示。

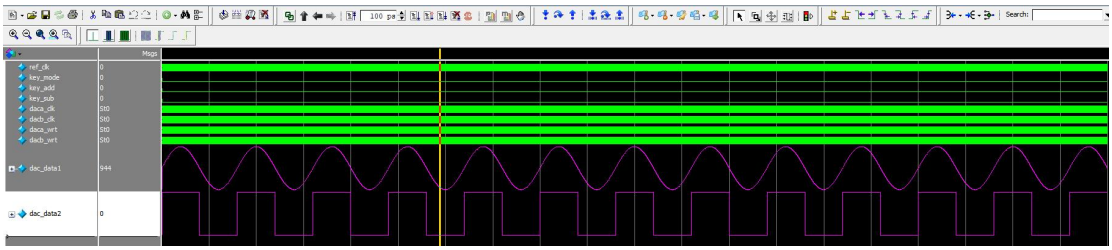


图 19-18 通道输出波形效果

19.6板级验证

19.6.1实验目标

本实验的板级验证环节，主要验证以下三个目标：

1. 能否正确的烧写和下载程序。
2. 下载并配置好输出波形后，能否在示波器上观察到期望的波形。
3. 按下设计好的频率、通道、波形类型相关按键，示波器上是否能发生频率、通道、波形类型等相关变化。

19.6.2系统所需硬件

系统所需硬件如下：

1. AC6103 开发板
2. 电源线一根
3. 下载器一个
4. ACM9767 模块
5. 示波器一台

19.6.3硬件连接

本次实验系统硬件方面连接十分简单，只需给 AC6103 开发板连接好电源线并将 ACM9767 模块连接到 AC6103 开发板的扩展接口上即可。ACM9767 模块的 CH1 与 CH2 通道则会按照用户的设置输出相应的波形，将通道与示波器相连即可看到产生的波形。当所有连接工作完成后将开发板的电源开关拨到 DC 侧，本次系统设计开发板连接方法如图 19-19 所示。

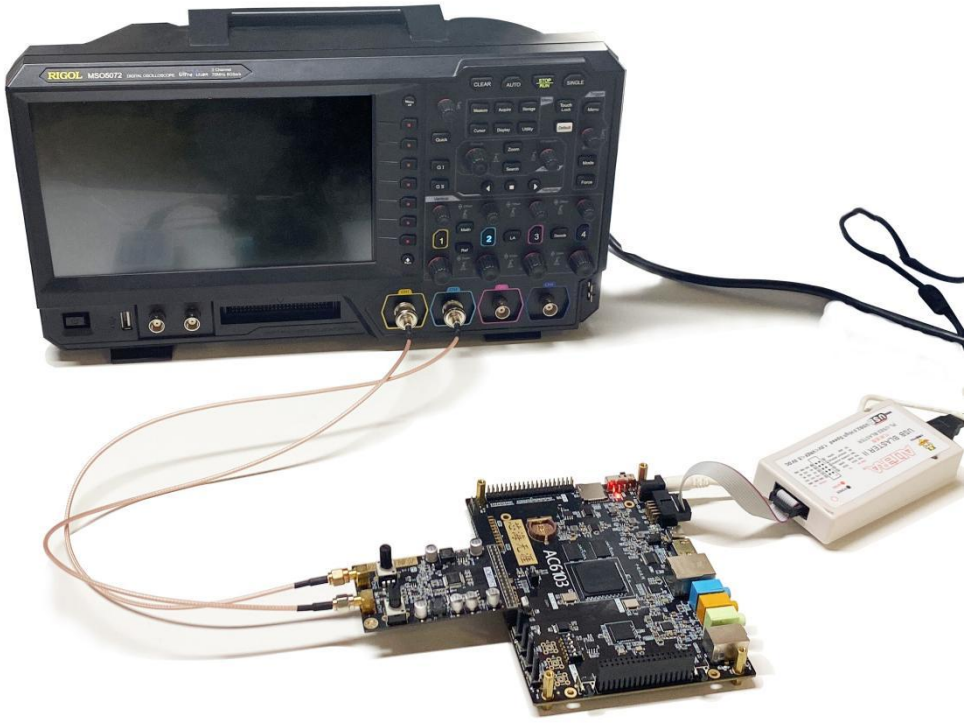


图 19-19 硬件连接图

19.6.4管脚绑定

前面我们也在整体设计阶段规划了按键控制功能。这里，在经过一系列设计与验证后，我们可以进一步深化确定硬件功能和相关参数。按设计，我们利用开发板上按键 S0 来切换工作模式，S1 控制加，S2 控制减，具体功能如下表

19-4 所示。

表 19-4 按键控制表

按键/拨码开关	功能描述
S0	切换工作模式
S1	在波形模式下，切换波形，在频率设置模式下，切换频率（增加）
S2	在波形模式下，切换波形，在频率设置模式下，切换频率（减小）

在 AC6103 开发板上挂载 ACM9767 模块，其管脚绑定对应关系如下表 19-5 所示：

表 19-5 管脚绑定表

功能信号	AC6103 开发板	功能信号	AC6103 开发板
Data0 [13]	PIN_A15	Data1[13]	PIN_F20
Data0 [12]	PIN_B14	Data1 [12]	PIN_D22
Data0 [11]	PIN_A16	Data1 [11]	PIN_F22
Data0 [10]	PIN_B15	Data1 [10]	PIN_H16
Data0 [9]	PIN_A17	Data1 [9]	PIN_G13
Data0 [8]	PIN_C17	Data1 [8]	PIN_H21
Data0 [7]	PIN_A18	Data1 [7]	PIN_J22
Data0 [6]	PIN_G16	Data1 [6]	PIN_H20
Data0 [5]	PIN_A19	Data1 [5]	PIN_K22
Data0 [4]	PIN_D19	Data1 [4]	PIN_H18
Data0 [3]	PIN_B19	Data1 [3]	PIN_L22
Data0 [2]	PIN_A20	Data1 [2]	PIN_K21
Data0 [1]	PIN_C19	Data1 [1]	PIN_K18
Data0 [0]	PIN_G17	Data1 [0]	PIN_J18
daca_clk	PIN_B22	dacb_clk	PIN_E21
daca_wrt	PIN_C21	dacb_wrt	PIN_D20
key_add	PIN_J8		
key_mode	PIN_B12		
Key_sub	PIN_L8		
ref_clk	PIN_G1		

19.6.5下载与验证

连接好硬件之后将开发板上的拨码开关拨到 DC 侧，将我们提供好的工程源码下载至开发板中即可从示波器中看到相应的波形。

下载完成后在示波器上显示的默认波形如下图 19-20 所示。

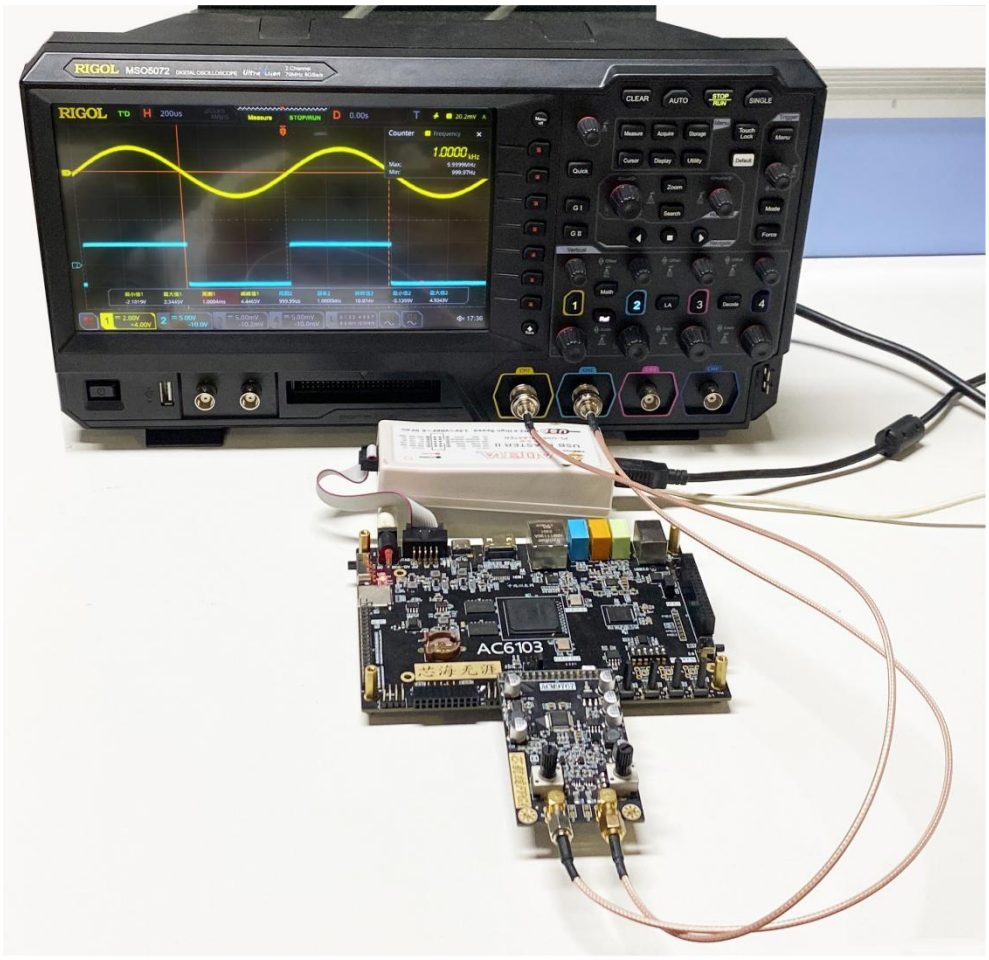


图 19-20 输出信号波形

由于我们默认的基准波形为 1KHz 的正弦波，示波器上显示也为 1KHZ，说明我们功能已经实现。

在随后的实验中，我们可以根据设计按下不同的按键，先按下按键 S0，进入 ACM9767 模块的 CH1 通道波形选择，然后按下按键 S1 来切换波形。接着按下按键 S0，进入 ACM9767 模块的 CH2 通道波形选择，然后按下按键 S2 来切换波形。示波器上显示波形如下图 19-21 所示。

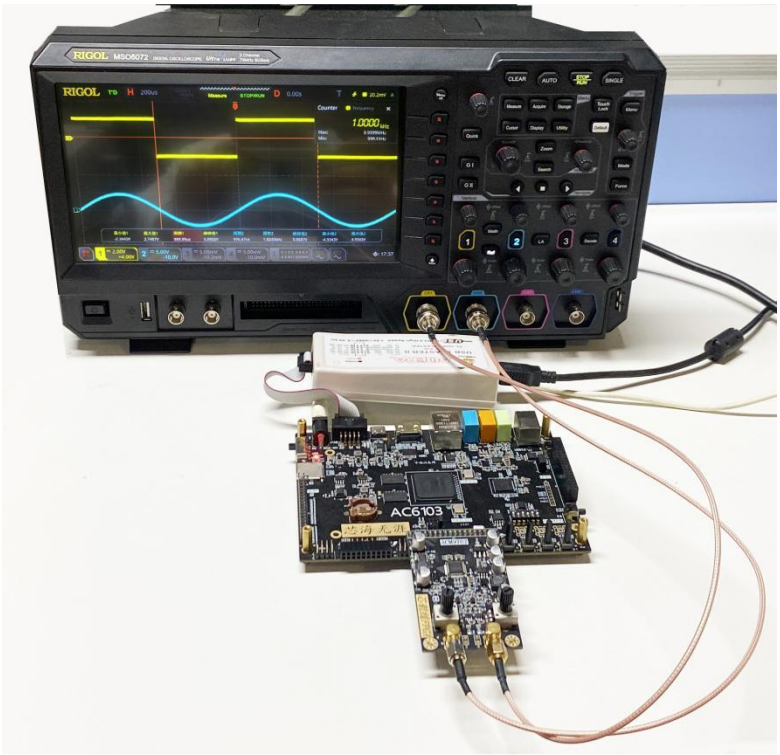


图 19-21 按下 S0、S1、S2 切换 CH1 和 CH2 输出波形

接着我们再按下按键 S0，然后按下按键 S1，改变输出频率，如下图 19-22 所示。

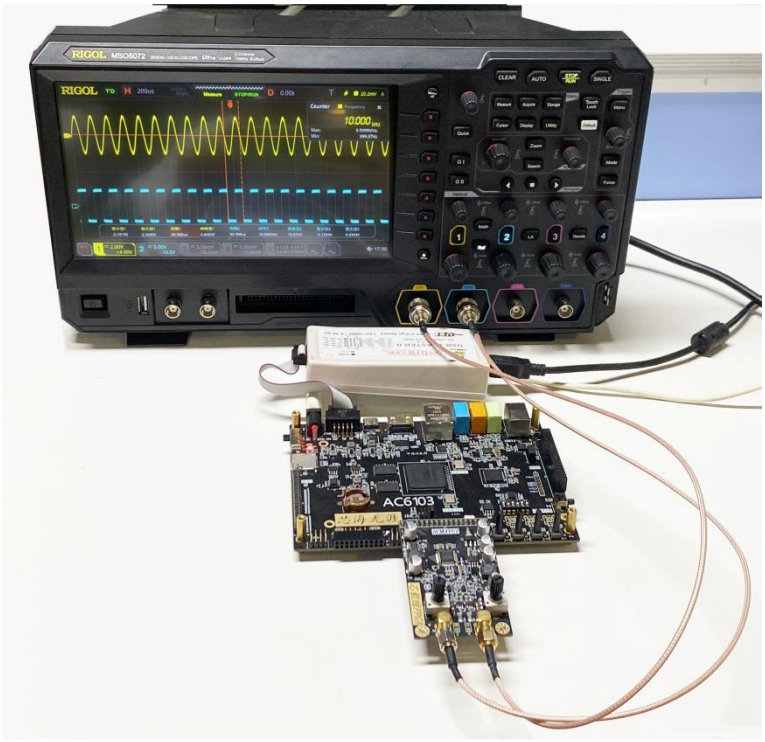


图 19-22 切换输出信号频率

通过上述板级验证，可以看到最终输出结果与我们设计需求一致，说明本次实验成功。

19.7常见问题与思考总结

1. 输出波形的类型，是通过 en_mode 信号实现的。为了实现输出波形的类型可选择，必须为每个通道开辟 3 个 rom 空间，用于存放代表 3 种不同类型的波形数值。
2. 本实验以 AC6103 开发板搭载 ACM9767 模块作为开发模型，以按键的跳变实现频率、波形类型和通道的档位切换。而真正在实现以此实验为基础的信号发生器开发时，可以考虑进行无极调节的硬件设计。借助带滑动变阻器的旋钮，频率和相位的控制将有更加丰富的选择，也更接近于一个真实的信号发生器。