实验九：波形发生器设计

1. 实验目的

1.1. 学习波形发生器的基本原理；

1.2. 基于先前实验的学习内容，例化多个模块并连线，实现一个波形发生器；

1.3. 学习多个模块的调试方法。

2. 实验内容

2.1. 使用 chisel 实现波形发生器，能够输出三角波、锯齿波、方波、正弦波；

2.2. 生成波形并通过 gtkwave 查看。

3. 实验相关 Chisel 语法介绍

线性反馈移位寄存器

如果要产生伪随机数，可以使用 chisel3.util.random 包里的线性反馈移位寄存器原语 LFSR：

```
def apply(width: Int, increment: Bool = true.B, seed: Option[BigInt] = Some(1)): UInt
```

第 1 个参数是寄存器的位宽。第 2 个参数是一个 Bool 类型的使能信号，用于控制寄存器是否移位，缺省值为 true.B。第 3 个参数是一个随机种子，是可选值类型。

它返回一个 UInt(width.W)类型的结果。例如：

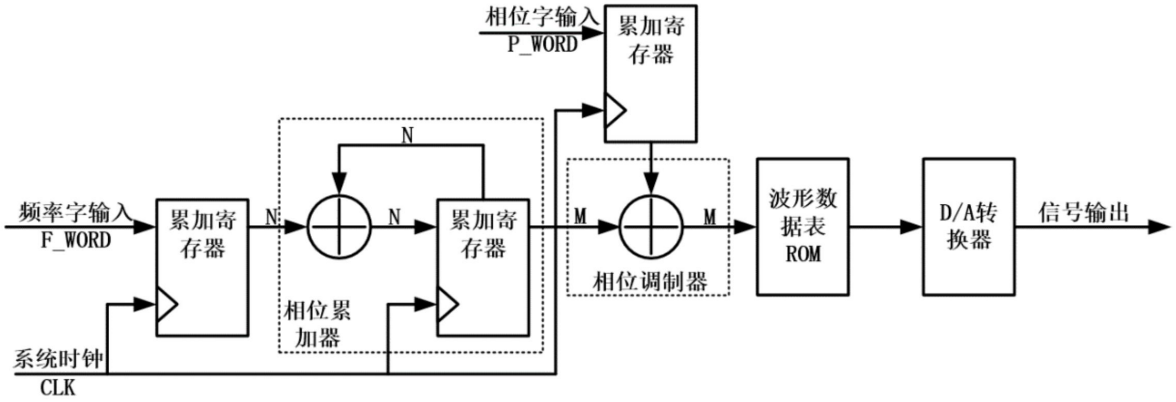
```
import chisel3._
import chisel3.util._

class LFSR16 extends Module {
  val io = IO(new Bundle {
    val en = Input(Bool())
    val out = Output(UInt(16.W))
  })

  io.out := LFSR(16,io.en,Some(1))
}
```

4. 实验步骤

波形发生器及其原理介绍

波形发生器是一种数据信号发生器，在调试硬件时，常常需要加入一些信号，以观察电路工作是否正常。

DDS（DirectDigitalSynthesis）是一种把一系列数字信号通过 D/A 转换器转换成模拟信号的数字合成技术。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，DDS 结构如下图所示。



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。（本次实验只用数字仿真的形式实现前三部分）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}。

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。

接下来介绍相位累加器如何得到相位码。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。

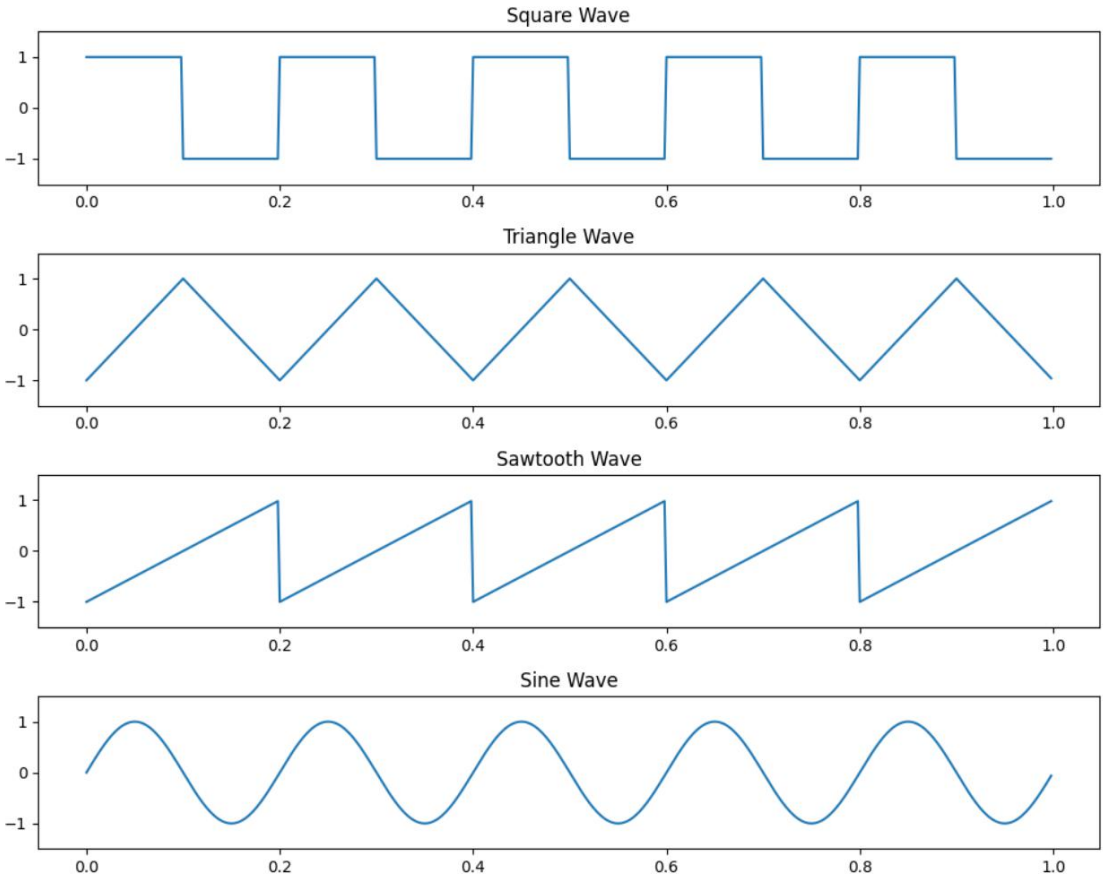
我们以频率控制字 K = 1 为例，相位累加器的低 20 位一直会加 1，直到低 20 位溢出向高 12 位进位，此时 ROM 为 0，也就是说，ROM 的 0 地址中的数据被读了 220 次，继续下去，ROM 中的 4096 个点，每个点都将会被读 220 次，最终输出的波形频率应该是参考时钟频率的 1 / 220，周期被扩大了 220 倍。同样当频率控制字为 100 时，相位累加器的低 20 位一直会加 100，那么，相位累加器的低 20 位溢出的时间比上面会快 100 倍，则 ROM 中的每个点相比于上面会少读 100 次，所以最终输出频率是上述的 10 倍。

本次实验只用数字仿真的形式实现前三部分，在 ROM 数据位宽为 8bit 的情况下，可以表示 0 到 255 共 256 个值，最终输出给 DA 转换器，DA 转换器的使用和配置与实际的芯片相关。我们这里可以简单理解为，如果电源电压为 xV，当 DA 转换器输入 0 时，DA 转换器输出 0V；当 DA 转换器输入 255 时，DA 转换器输出 xV，其余输入会等比映射到 0~xV 的输出。

任务一：

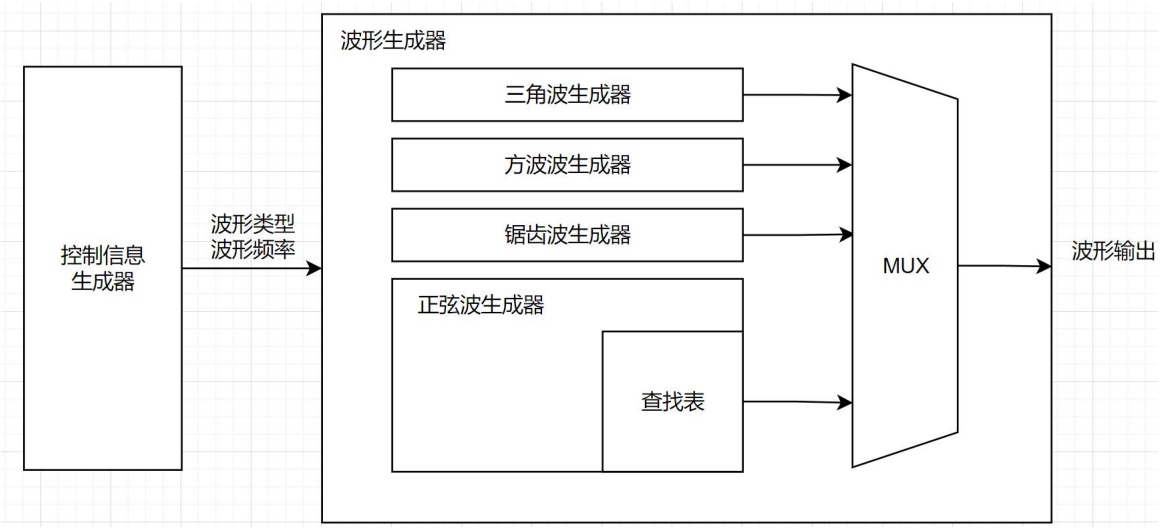
据上述对波形发生器的介绍，实现相位累加器、相位调制器、波形存储器这三个部分，使得能够输出基本的方波（Square Wave）、三角波（Triangle Wave）、锯齿波（Swatooth Wave）和正弦波（Sine Wave）（如下图所示），并且自定义一个控制端口，能够通过外部信号来控制具体

的波形类型。；



- 假设输入的始终频率不变，ROM 存储单元的个数为 512，每个数据用 8 位 2 进制表示，即 ROM 地址线宽度为 9，数据线宽为 8。；使用 ChiselTest 进行仿真测试并生成 VCD 波形。；需要能够使用 gtkwave 观察到不同的波形（需要使用 Analog 的方式来查看相关信号，具体请查看实验六中的相关介绍）。
- 波形的频率可设置。(我们定义 256 个时钟周期为一个基准周期，称为基准周期 T，对应的基准频率为 f，我们需要输出波形频率为 f 和 2f 的波形。
- 分离控制模块和波形发生器模块，两个模块之间使用 valid 和 ready 信号进行通讯。波形发生器的 ready 信号会在一定时间内拉低（可以使用前面介绍的 LFSR），当 ready 信号拉低时无法接收控制模块的控制信息。

实验要求和提示：



1. 整个模块层级如上图所示，分为控制信息生成器和波形生成器两部分（在代码中已经写好了顶层模块接口，学生只需要实现模块中的逻辑）

2. 控制信息生成器（DDSCtrl）输出频率信息和波形类型（波形类型有 4 种，代码中分别表示三角波、锯齿波、方波和正弦波；频率信息有 2 种，f 表示基准频率，f_2 表示 2f 频率，已经定义好了变量名，可以直接使用）

```
object WaveType {
  val triangular = 0.U
  val sawtooth = 1.U
  val square = 2.U
  val sine = 3.U

  val width = 2

  def apply(): UInt = UInt(width.W)
}

object FrequencyType {
  val f = 0.U
  val f_2 = 1.U

  val width = 1
  def apply(): UInt = UInt(width.W)
}
```

3. 波形生成器（DDSGen）中有 4 个小模块，其中的三角波生成器（代码中 TriangularWaveGen）在实验六中已经实现过；方波（SquareWaveGen）和锯齿波（SawtoothWaveGen）的生成与三角波（TriangularWaveGen）十分类似，只需要更改一下计数器计数的最大值即可。正弦波生成器

（SineWaveGen）首先需要生成地址，去读查找表，通过地址的跨步来控制波形的频率（类似实验六的做法）。其中查找表的内容已经提供（SineDataRAM），只需要通过地址进行读取即可。

4. 最后需要通过一个 4 选 1 的选择器输出需要的波形。

5. 需要通过 gtkwave 分别展示（截图）4 种波形（频率分别为 f、2f）共 8 种组合情况。

6. 代码中顶层端口的 flag 信号每隔 256 个时钟周期就会拉高一次，这样我们可以方便得看出我们实现的波形是否正确，例如下图为输出正弦波，频率为 f 的截图。