**实验报告**

专业： 混合班

姓名： 许展风

学号： 3210100658

日期： 3月22日

地点： 东4-223

课程名称： 数字系统设计实验 指导老师： 屈民军 成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验名称： Lab19 音乐播放实验 实验类型： 设计型

1. 实验目的
2. 掌握音符产生的方法，了解DDS技术的应用。
3. 了解音频解码的应用。
4. 掌握系统“自顶而下”的数字系统设计方法。
5. 实验任务与要求

设计一个音乐播放器，要求以下条件：

1. 可以播放4首歌曲，设置play/pause\_button、next\_button、reset三个按键。按play/pause\_buttun键，音乐在播放和暂停之间切换；按next\_buttun键播放下一首乐曲。
2. LED0指示播放情况（播放时点亮）、LED2和LED3指示当前乐曲序号。
3. 实验原理与设计
4. 电路的总体设计

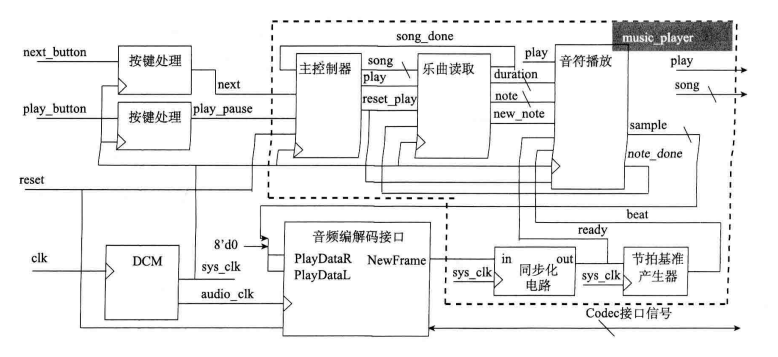
系统由时钟管理模块（DCM）、按键处理模块、音频编解码接口模块、次顶层音乐播放模块（music\_player）组成。

DCM模块产生频率为100MHz的系统时钟和12.5MHz的音频时钟。

按键处理模块对play\_button和next\_button两个按键信号进行处理，使得按下按钮后，输出一个时钟周期的高电平脉冲。

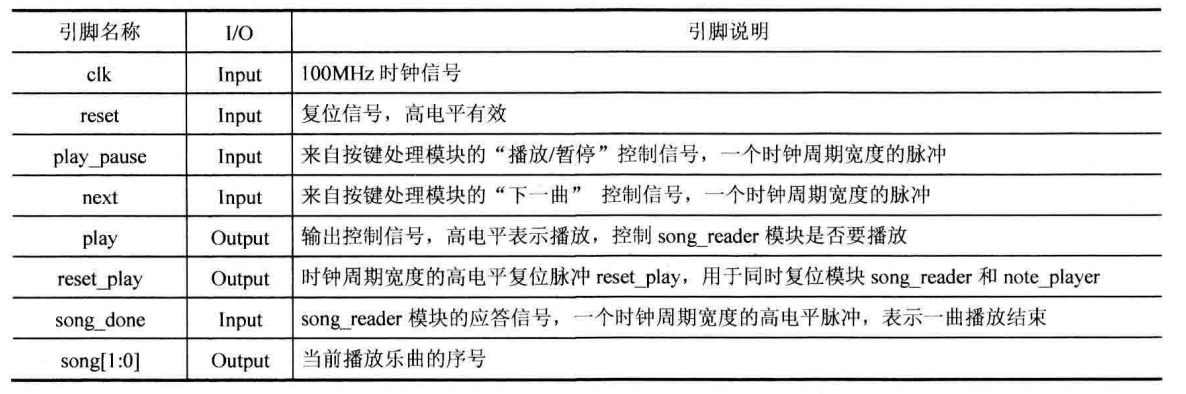
音频编解码接口模块收取音符的正弦波样品，转换为串行输出并发送给音频解码芯片ADAU1761。芯片将收到的正弦波样品进行AD转换并放大，最后送到扬声器播放。这一模块由网表文件的形式直接提供。

music\_player模块接收按键信息，输出播放状态信号，和正弦波样品信号，它是整个系统中的核心控制模块。它主要分为主控制器、乐曲读取、音符播放、同步化电路和节拍基准产生器五个子模块。

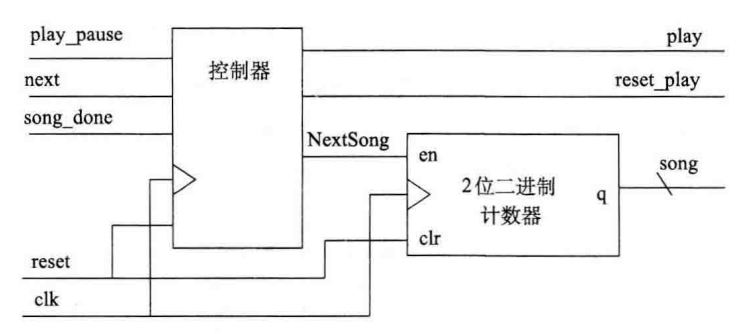


1. 子模块设计
   1. 主控制模块mcu设计

主模块mcu的目标功能是响应按键信息，由此控制系统播放、重置状态与乐曲的选择状态。因此模块的输入是时钟、按键响应信号，来自下一级系统的音乐播放完成反馈信号，输出控制信号以及乐曲的选择状态信号。具体端口如下：



模块的状态控制信号处理可以通过状态机来实现，乐曲状态的设置，由于只有一个按键信号，所以可以通过使能计数器的方式来实现。由此设计主模块mcu的结构框图。



控制器的状态设计如下图。设置四个状态，三个输出。

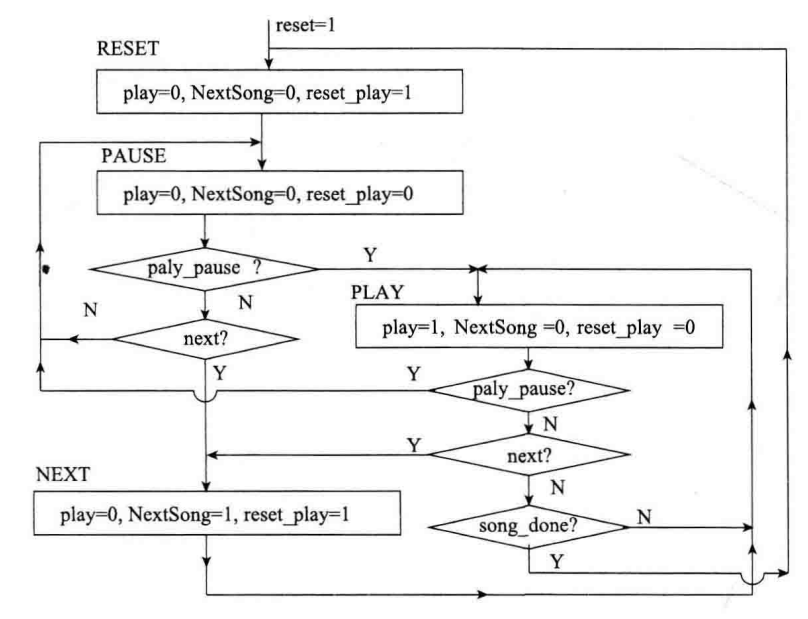
复位信号reset信号控制进入RESET状态，输出reset\_play=1，控制后面的模块初始化，并直接进入PAUSE状态。

PAUSE状态等待控制信号，输出都是0，如果play\_pause为1，则进入PLAY状态；如果next为1，进入NEXT状态；否则保留原状态。

PLAY状态为播放状态，输出play = 1，如果play\_pause为1，则回到PAUSE状态；如果next为1，进入NEXT状态；如果song\_done为1，说明一首乐曲播完，进入复位RESET状态；否则保留原状态。

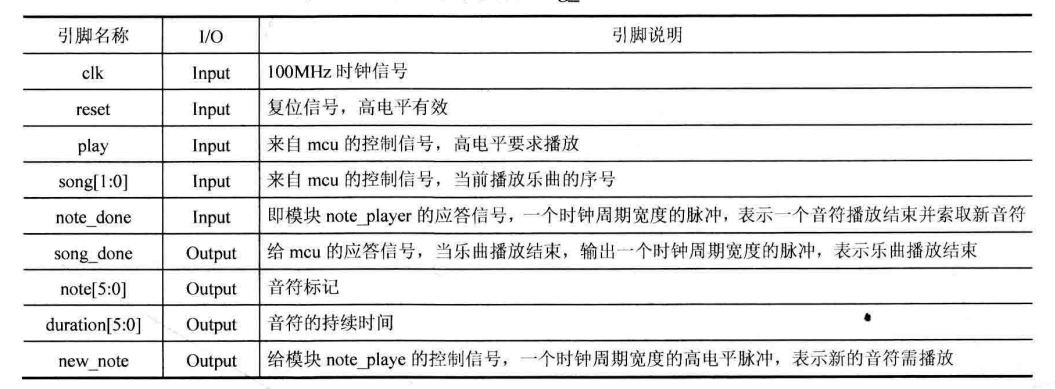
NEXT状态，一要乐曲序号加一，二要复位后面的模块，因此输出nextsong为1，使得计数器加1，输出reset\_play为1。之后要直接播放下一曲，因此直接进入PLAY状态。

ASM图如下：

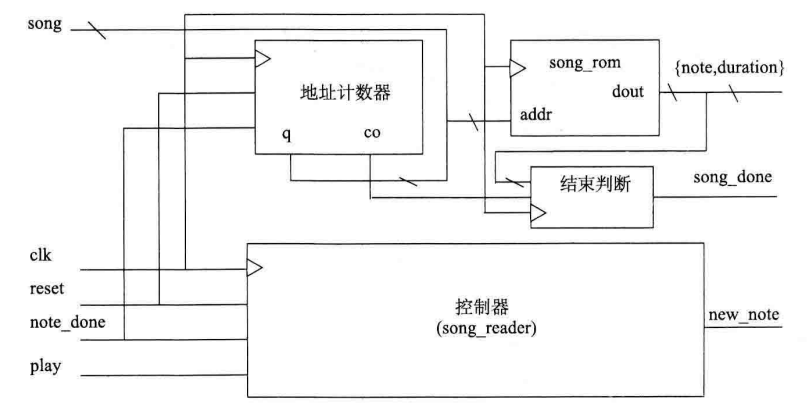


* 1. 乐曲读取模块song\_reader设计

该模块的功能是，播放时将此时应该播放的音符取出并送给下一各模块，播放的几首歌曲由控制器mcu模块控制，什么时候取出新音符由下一个模块控制。每首歌会有最多32个音符，要判断一首乐曲是否播放完，并告知控制模块。因此它的输入输出、以及端口连接都很清楚。



该模块依旧需要一个控制器来控制播放与停止。song\_rom模块中存放四首歌曲，每首歌有32个音符，输入地址即可从中取出音符，因此需要一个5位的递增计数器给出递增的音符地址，高位加上song这个歌曲序号即可作为输入地址。还需要一个结束判断的模块，计数器的进位以及音符的duration=0，表示一曲结束。由此设计结构框图：



音符的播放过程可以分为未播放、播放中两个状态，由于note\_done信号为1后，需要两个时钟周期后，dout端的数据才是新的音符，所以new\_note的响应也应在新数据输出后置一，所以需要多一个状态实现一个时钟周期的延迟，加上复位状态一共四个状态，一个输出。

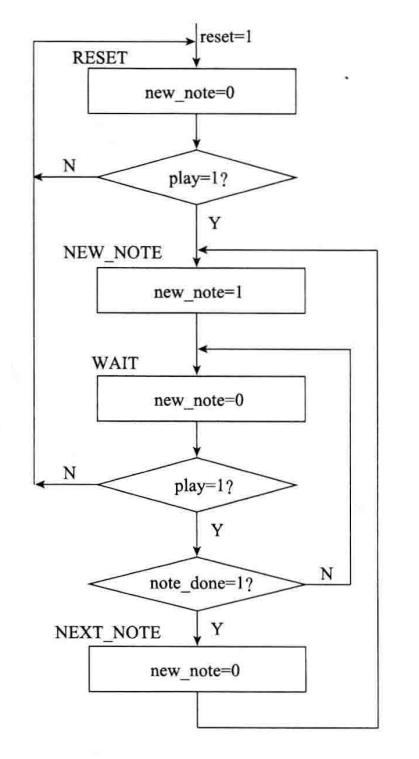
复位信号reset信号控制进入RESET状态：输出new\_note=1，如果play=1，进入NEW\_NOTE状态，否则回到RESET。

NEW\_NOTE状态，输出new\_note=1，直接进入WAIT状态。

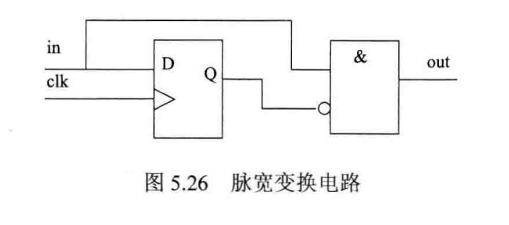
WAIT状态，new\_note=1，如果play=0，回到RESET状态，play=1有效，检查note\_done，如果为0，保持WAIT状态，如果为1，进入NEXT\_NOTE状态。

NEXT\_NOTE状态，new\_note=0，直接进入NEW\_NOTE状态。

ASM图如下：

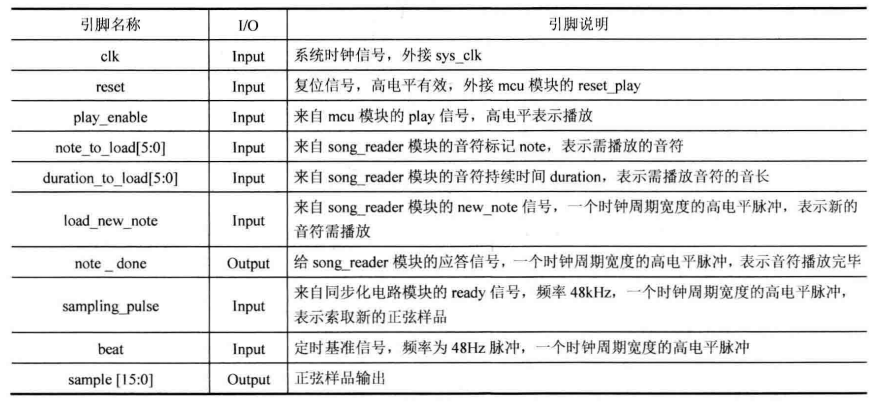


结束判断模块，如果co=1，或者，duration为0，则输出一个时钟周期的高电平脉冲，由于duration的脉冲宽度可能时间很长，本实验中采用脉宽变换电路实现：



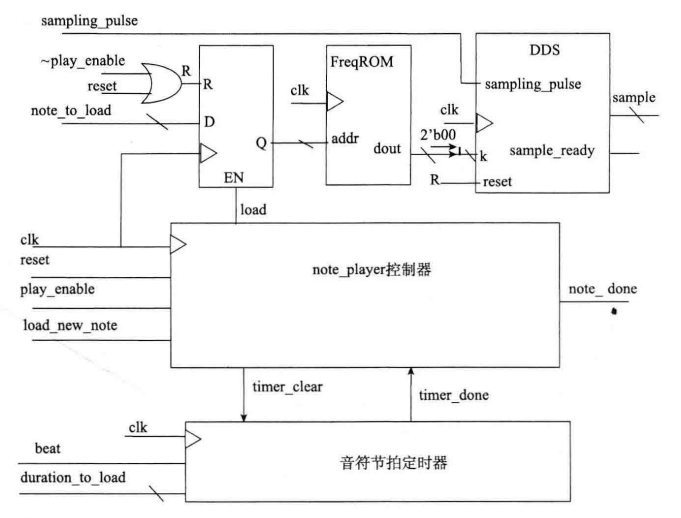
* 1. 音符播放模块note\_player设计

音符播放模块将根据收到的音符（包括音高和音符时间）据此确定DDS的相位增量k。DDS的采样脉冲由同步化电路模块产生，音符的播放节奏需要由节拍信号控制。输出正弦信号样品和音符播放完成的信号，由此确定模块的接口：



该模块由一个控制器控制时序，一个5位D寄存器存储收到的音符频率，根据控制器的信号进行更新，将输出给FreqROM模块，输出对应的k值，高位加00后输出给DDS模块。

而音符播放时间的控制，需要音符节拍定时器，根据收到的音符时间进行定时，是一个6位二进制计数器，节拍beat作为它的使能信号，timer\_clear作为复位信号，timer\_done作为进位输出信号。由此得到结构图如下：



控制器的状态也有四个，复位、等待、播放完毕、新音符播放状态，三个输出，一是timer\_clear复位音符节拍定时器，二是load使能输入新的音符，三是note\_done标志音符播放完毕。

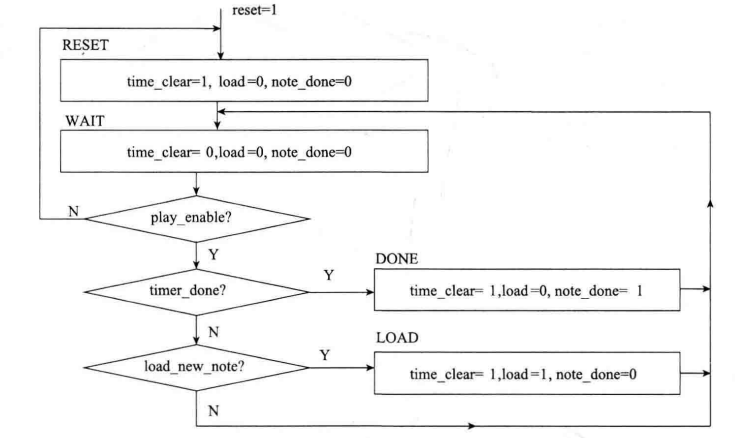
复位信号reset信号控制进入RESET状态，输出time\_clear=1，复位计时器，进入WAIT状态。

WAIT状态，三个信号都不是高电平输出，判断play\_enable信号，为0，返回RESET状态；为1，判断timer\_done信号，为1，进入DONE状态；为0，判断load\_new\_note信号，为1，进入LOAD状态，为0，进入WAIT状态。

DONE状态，输出timer\_clear=1，note\_done=1，直接进入WAIT状态。

LOAD状态，输出timer\_clear=1，load=1，直接进入WAIT状态。

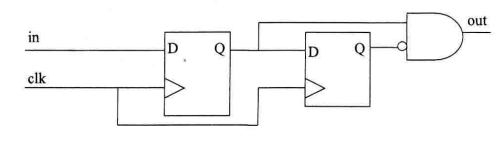
ASM图如下：



* 1. 同步化电路

由于音频编解码接口模块与其他模块采用不同的时钟，audio\_clk和sys\_clk，两者之间

的控制及应答信号需要同步化处理。音频编解码接口模块输出信号NewFrame脉冲宽度为一个audio\_clk的时钟周期，和sys\_clk输入电路后，输出一个与sys\_clk同步，且脉冲宽度相同的信号ready，它可以作为DDS的采样脉冲，和节拍基准产生器的使能分频信号。



* 1. 节拍基准产生器

输入48kHz的ready信号和时钟信号，要产生48Hz的节拍定时基准脉冲信号，因此它是一个分频比1000的分频器。

1. 核心模块HDL代码
   1. Mcu控制器模块代码

本实验的很多模块都需要控制器，需要用状态机方法实现，在本实验中均采用两端式的写法，根据ASM图，可以很公式化的编写，需要注意的是state、nextstate的reg数据类型以及数据位数，在状态转换的if判断时，要把所有情况都考虑到并编写入代码中。状态机方法过于公式化，因此只以mcu控制器为例。

//状态编码

parameter RESET = 0; //初始复位，直接进入PAUSE

parameter PAUSE = 1; //暂停状态，响应play\_pause, next

parameter PLAY = 2; //播放状态，响应play\_pause, next, song\_done

parameter NEXT = 3; //下一首，直接进入PLAY

reg [1:0] state,nextstate;

//时序电路

always @(posedge clk) begin

if(reset) state = RESET;

else state = nextstate;

end

//组合电路

always @(\*) begin

play = 0;

NextSong = 0;

reset\_play= 1;

case (state)

RESET:begin

nextstate = PAUSE;

end

PAUSE:begin

if(play\_pause) nextstate = PLAY;

else if(next) nextstate = NEXT;

else nextstate = PAUSE;

reset\_play = 0;

end

PLAY:begin

if(play\_pause) nextstate = PAUSE;

else if(next) nextstate = NEXT;

else if(song\_done) nextstate = RESET;

else nextstate = PLAY;

play = 1;

reset\_play = 0;

end

NEXT:begin

nextstate = PLAY;

NextSong = 1;

End

//状态编码没有用完所有state的位数，case需要多加一个default进行错码处理

endcase

end

* 1. 音符播放模块的音符节拍定时器模块

在之前的实验中所编写的定时器模块，定时的时间是由参数控制的，而这里定时的时间由变量端口输入。

parameter counter\_bits = 1;

input r;

input clk;

input en;

input [5:0] duration;

output co;

reg [counter\_bits-1:0] q;

assign co=(q==(duration-1))&&en;

always @(posedge clk) begin

if(r) q=0;

else begin

if(en) begin

if(q==duration-1) q=0;

else q = q + 1;

end

end

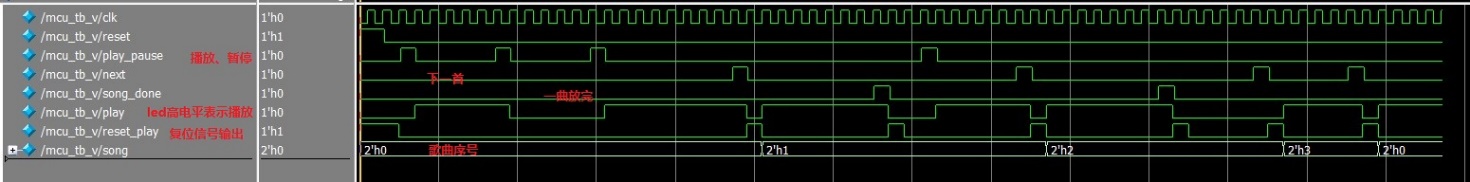
end

endmodule

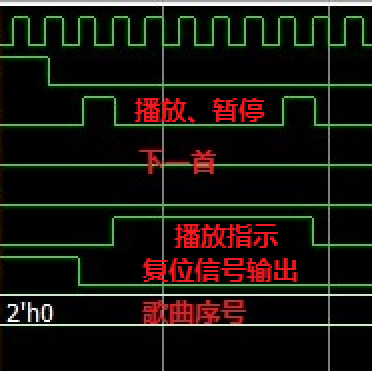
1. 主要仪器设备

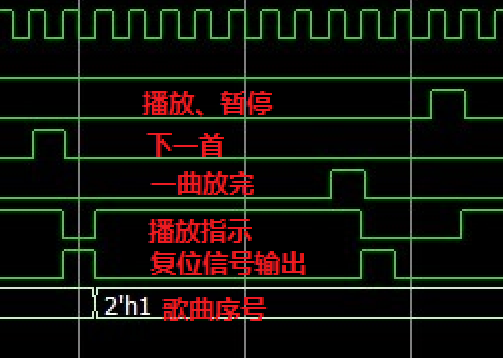
装有Vivado、ModelSim SE软件的计算机，Nexys Video开发板，耳机。

1. 实验步骤
2. 建立实验文件夹，将文件夹分类管理。
3. 编写mcu模块的Verilog HDL代码，并用ModelSim仿真验证。
4. 编写song\_reader模块的Verilog HDL代码，并用ModelSim仿真验证。
5. 编写note\_player模块的Verilog HDL代码，并用ModelSim仿真验证。
6. 编写同步化电路的Verilog HDL代码及其测试代码，并用ModelSim仿真。
7. 编写次顶层music\_player模块的Verilog HDL代码及其测试代码。
8. 新建music\_player的Vivado工程，编写顶层music\_player\_top模块的Verilog HDL代码，生成符合要求的DCM内核，添加music\_player模块及其子模块、音频编解码接口模块的网表文件AudioInterface.edf和端口文件AudioInterface.v、按键处理模块提供综合网表文件button\_press\_unit.edf和端口文件button\_press\_unit.v、时钟和引脚约束文件MusicPlayer.xdc等。对工程进行综合、约束、实现，并下载工程文件到Nexys Video开发板中。
9. 将耳机接入实验开发板音频输出插座，操作reset（中间按钮）、play/pause（右边按钮）、next（下面按钮）三个按键，试听耳机中的乐曲并观察实验板上指示灯变化情况，验证设计结果是否正确。
10. 实验结果和分析处理
11. 仿真波形分析
    1. mcu模块仿真

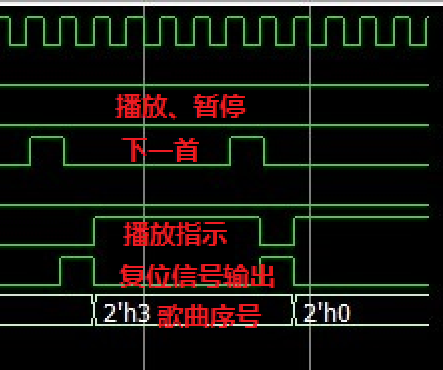


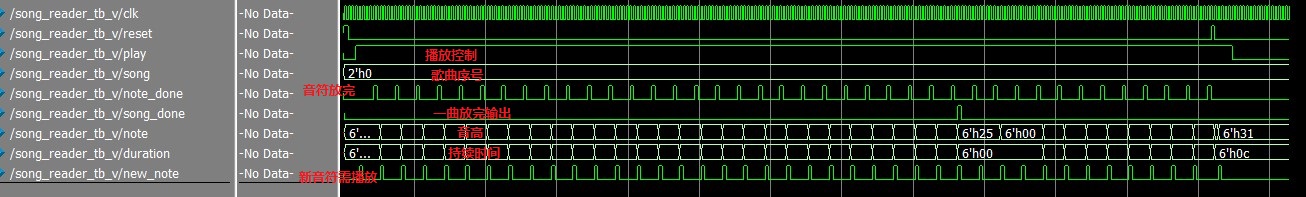
本模块的输入控制信号是play\_pause和next，song\_done，输出是play，reset\_play，song三个信号，分析是主要观察控制信号的变化对应的输出是否正确。



* + 1. 局部放大图1：首先复位系统，song\_done、play输出为0，歌曲序号为0，系统输出复位信号为1。播放、暂停键按下后，给系统一个高电平脉冲，play输出为1，reset\_play为0，再次按下，输入高电平脉冲后，音乐暂停，play输出为0。该部分仿真正确。
    2. 局部放大图2：下一首键按下后，给系统一个高电平脉冲，系统play=0，reset\_play=1，进行复位，一个时钟周期后，歌曲序号加一，play=1，reset\_play=0，播放下一首歌曲。

Song\_done信号也是一个高电平脉冲，表示一曲放完，输入系统后，停止播放，输出一个时钟周期复位信号，播放、暂停键再次按下，系统继续播放。该部分仿真正确。

* + 1. 局部放大图3：在原本暂停播放的状态下，按下下一首按键，同样进行一个时钟周期的复位后，play=1，歌曲序号为3，播放第三首歌，再次按下后，歌曲序号变为0。该部分仿真正确。
  1. song\_reader模块仿真

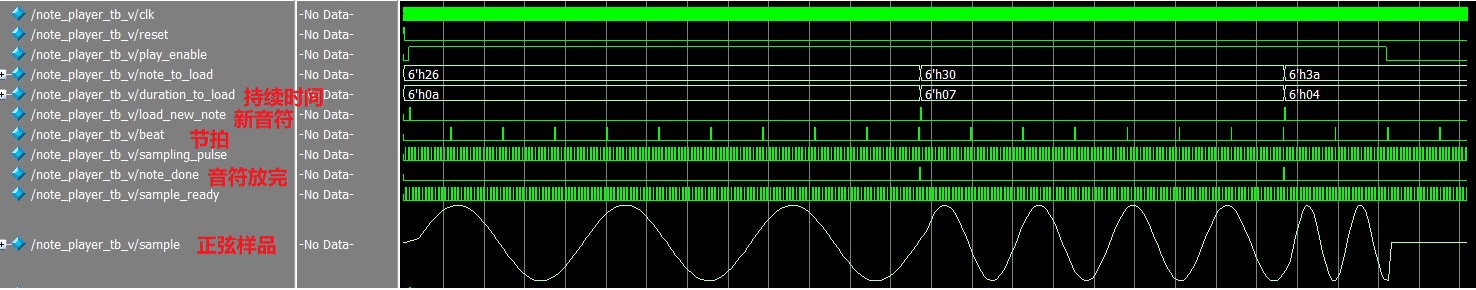




* + 1. reset复位后，系统读入第一个音符，play信号置1后，系统开始播放，new\_note输出高电平脉冲，告知下一模块新音符需播放，收到note\_done高电平脉冲，说明音符播放完，下个时钟周期读入新的音符，并再输出new\_note高电平脉冲。该部分仿真正确。

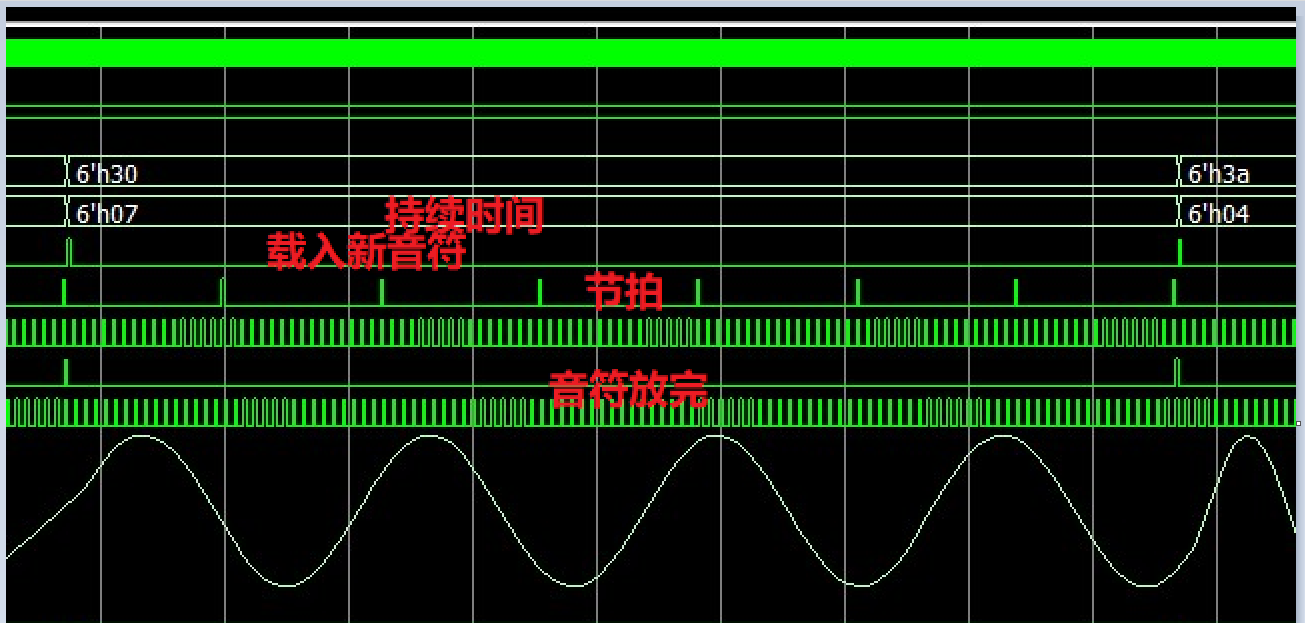


* + 1. 当读取到的duration为0时，song\_done输出一个高电平脉冲，表示一曲放完。该部分仿真正确。
  1. Note\_player模块仿真

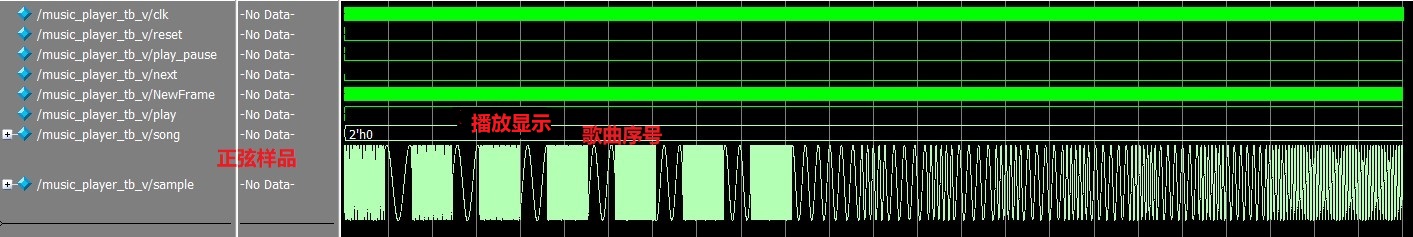


本模块仿真通过输入的音符，播放出正弦样品，需要检查音符播放的持续时间，检查播放频率是否正确变化，控制信号时序是否正确。

从完整信号图中可以看到随着音符音高的更新，正弦样品的频率也发生变换，play\_enable置0后，正弦样品也停止输出。

通过定时器进位，判断音符播放完毕，输出note\_done高电平脉冲表示音符放完，音符在上级模块更新后，改模块收到load\_new\_note信号，播放新的正弦样品，持续时间与节拍的间隔数量是一致的。如图中duration=7，节拍走过7个间隔，输出note\_done高电平。该部分仿真正确。

* 1. Music\_player模块仿真



按下播放键后，play输出指示置高，正弦样品开始输出，频率随着音符变化而变化。仿真结果正常。

1. 讨论、心得
   1. 实验问题反思：

实验中的困难主要在仿真调试与硬件调试方面，在电路设计和代码编写方面，跟着实验教程的思路逐步学习，在调试中检查出一些基础的语法错误与疏漏，整体的源文件代码都是能较为容易地编写的。

在仿真调试中，主要困难在于结束判断模块的调试修改，同时也是代码设计上的问题。一开始采用了状态机的方法去设计模块，根据co或duration==0的判断，输出一个时钟周期的高电平脉冲，设计了RESET，PLAY，ENDING三个模块，采用了两段式的方法编写，但modelsim调试的波形结果检查发现，输出的song\_done信号会滞后一个时钟周期。可能的原因是时序设计的问题，也可能是编写的问题，我参考过其他同学的有效的状态机方法写的代码，并不是严格的两端式。最终我选择使用了脉宽变换电路来直接实现，只需要一个D寄存器和逻辑语句，更加简洁直接。

在硬件上电调试时，很意外的会遇到很多问题。本来一直认为仿真波形没有问题，上电应该也不会有太多的错误。而事实上，从无音、单音到不能停止的问题我都遇到过。由此才懂得仿真的测试只是最基本功能的调试，而在最终上电时，有关端口之间连接的问题才会暴露出来。而在处理这个问题上，去分析RTL级综合的原理图就更加具有针对性。实验中的主要问题是music\_player次顶层模块编写时，ready信号没有接到note\_player模块，这个问题通过一步步深入的仿真调试解决了，另一个问题是在初次问题调试时将mcu控制器处的song\_done置0了，而最后调试时，我只检查到mcu模块，以为song\_done接回去了，事实上在模块中断开的，这个问题，在老师的帮助下通过RTL级综合的原理图才发现并解决。

在几个问题的调试过程中，我对状态机的HDL编写、时序逻辑的设计有了更深一步的理解，同时也逐渐学会了仿真波形调试，RTL级综合原理图调试的方法，都是从问题发生处出发，往临近的方向去逐一摸排，不能太想当然认为某些地方一定没问题，需要更加严谨，也需要对理论设计有一定程度的把握。

* 1. 在整个实验的过程中，在知识原理层面对DDS、音乐播放原理有了一定程度的掌握，在代码编写层面，对顶层模块的编写更加熟练，对状态机二段式的写法更加熟悉，也稍微学习了testbench的编写方法。在调试方法与经验上收获的更多。整个实验过程，更多的还是学习过程，一步步跟着设计流程去模仿实现与学习。如果要更进一步的创新层面，可以尝试去从播放转到录音，或是在数据层面上，对音乐进行适当的处理，在乐曲的选择上，拓展更多的功能，如顺序选择、单曲循环、随机选择等等，在播放上，去实现快放、慢放，这些都可以在本实验的基础上进一步对相关内容做拓展。

1. 思考题
   1. 在实验中，为什么next\_button、play\_pause\_button两个按键需要消颤动及同步化处理，而reset按键不需要消颤动及同步化处理？

答：两者的有效触发情况是不一样的。reset是高电平触发复位，系统复位后，reset无论为0或1都对系统无影响，所以不需要做消颤动处理。而next、play两个按键信号的有效情况是状态机方式实现的，需要一个时钟周期的高电平脉冲，如果该信号有过多毛刺，且时间间隔长，不同步，很可能使得系统状态发生错误的改变，而不能正常工作。

* 1. 在主控制器（mcu）设计中，是否存在接受不到按键信息？若存在，概率多大？有没有必要修改设计？

答：存在。设计中RESET状态直接进入PAUSE状态，NEXT状态直接进入PLAY状态，这两步都没有判断按键的输入信号，如果处于这两个状态时，进来输入信号，将会接收不到按键信息。但这两个状态都只持续一个时钟周期，在整个过程中时间很短，而且正常播放时，较少情况下会进入这两个状态，不会对系统功能实现有大的影响，没有必要修改设计。