



本科实验报告

课程名称： 数字系统设计实验

姓 名： 袁东琦

学 院： 信息与电子工程学院

系： 信电系

专 业： 信息工程

学 号： 3200103602

指导教师： 屈民军/唐奕

2022 年 6 月 1 日

专业: 信息工程
姓名: 袁东琦
学号: 3200103602
地点: 东四 223

浙江大学实验报告

课程名称: 数字系统设计实验 指导老师: 屈民军/唐奕

实验名称: 音乐播放实验

一、实验目的

- 掌握音符的产生方法，了解 DDS 技术的应用。
- 了解音频编解码的应用。
- 掌握系统“自顶而下”的数字系统设计方法。

二、实验任务

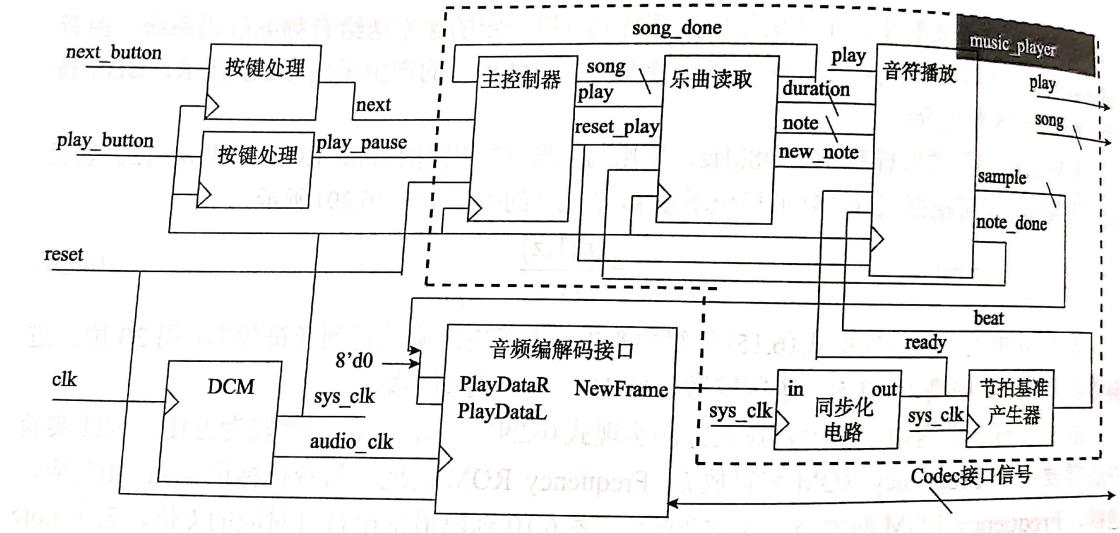
设计一个音乐播放器，要求以下条件。

(1) 可以播放四首乐曲，设置 play/pause_button、next_button、reset 三个按键。按 play/play_pause 键，音乐在播放和暂停之间切换；按 next_button 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。

三、实验原理

根据实验任务可将系统划分成时钟管理模块 (DCM)、按键处理、主控制器、乐曲读取、音符播放 (note_player)、同步化电路、节拍基准产生器和音频编解码接口电路等子模块，如下图所示。各主要子模块作用如下。



时钟管理模块 (DCM) 产生 100MHz 的系统时钟 sys_clk 和 12.5MHz 的音频时钟 audio_clk。

主控制器 (mcu) 模块接收按键信息，通知 song_reader 模块是否要播放 (play) 及播放哪首乐曲 (song)。

乐曲读取 (song_reader) 模块根据 mcu 模块的要求，逐个取出音符信息 {note,duration} 送给 note_player 模块播放，当一首乐曲播放完毕，回复 mcu 模块乐曲播放结束信号 (song_done)。

音符播放接收到需要播放的音符，在音符的持续时间内，以 48kHz 速率送出该音符的正弦波样品给音频编解码接口模块。当一个音符播放结束，向 song_reader 模块发送一个 note_done 脉冲索取新的音符。

音频编解码接口模块负责将音符的正弦波样品转换为串行输出并发送给音频编解码芯片 ADAU1761。

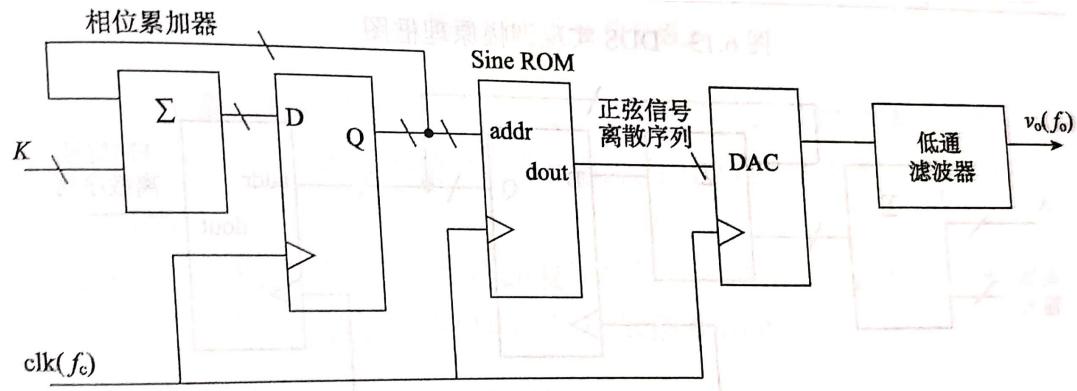
音频编解码芯片 ADAU1761 接收正弦波样品，在进行 AD 转换并放大，最后送至扬声器播放。

由于音频编解码模块与系统使用不同时钟，因此需要同步化电路协调两部分电路。

节拍基准产生器 48Hz 的节拍定时基准脉冲信号 (beat)，而 ready 信号频率为 48kHz，因此，节拍基准产生器为 1000 分频器。而按键处理模块完成输入同步化、放颤动和脉宽变换等功能。

1、DDS 的设计、

DDS 的基本原理框图如下图所示，由相位累加器。正弦查询表（Sine ROM）、D/A 转换器和低通滤波器组成。

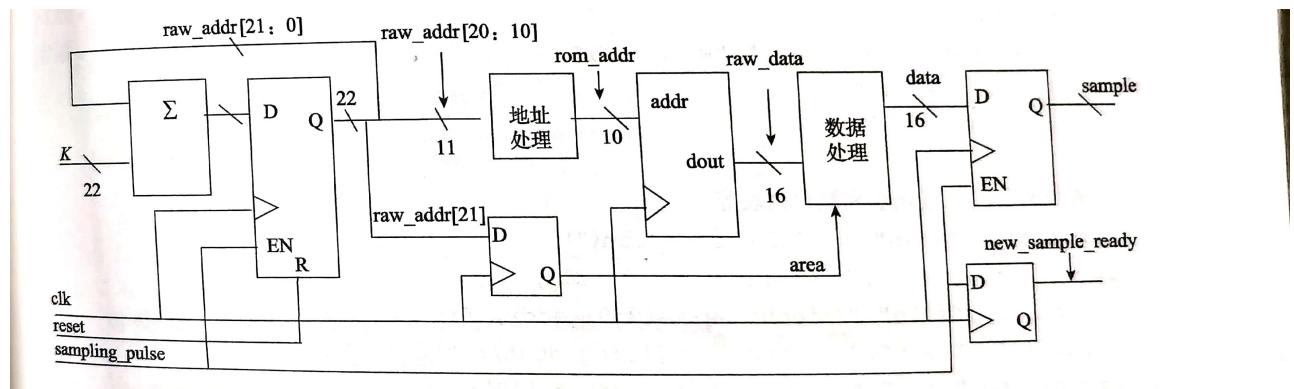


(1) 优化构想

优化的目的是，在保持 DDS 原有优点的基础上，尽量减少硬件复杂性，降低芯片面积和功耗，提高芯片速度。

(2) 优化后 DDS 的结构

因为必须利用 1/4 周期的 1024 个样品复制出一个完整正弦周期的 4096 个样品，所以对 DDS 结构进行必要处理，优化后的 DDS 结构如下图所示。



在很多应用场景，时钟信号与采样脉冲不是同一信号。Sampling_pulse 为采样脉冲，一般情况下，采样脉冲的频率应低于时钟 clk 的频率，而 sampling_pulse 宽度必须为一个时钟周期。若采样脉冲与时钟信号为同一个信号，可让 sampling_pulse 接高电平即可。

为了提高工作速度，采用了流水技术。

相位累加得到 22 位原始地址 $raw_addr[21:0]$ ，整数部分 $raw_addr[21:0]$ 为完整周期正弦信号样品的地址，其中高两位地址 $raw_addr[21:20]$ 可区分正弦的四个区域。由于 sine_rom 只保存了 1/4 周期的 1024 个样品，因此 $raw_addr[21:10]$ 不能直接作为 sine_rom 地址，必须进行必要的处理。处理方法如下所示。

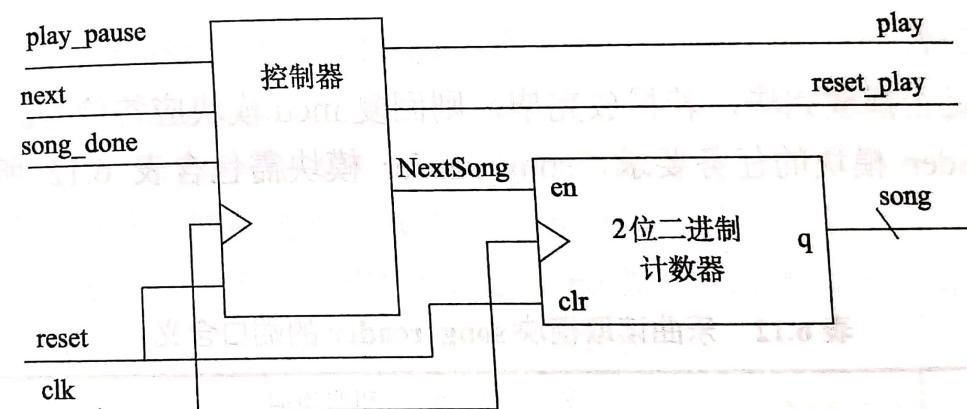
正弦区域	Sine ROM 地址	正弦样品 data	备注
0	$raw_addr[19:10]$	$raw_data[15:0]$	
1	当 $raw_addr[20:10]=1024$ 时，rom_addr 取 1023， 其他情况取 $\sim raw_addr[19:10]+1$	$raw_data[15:0]$	地址镜像翻转
2	$raw_addr[19:10]$	$\sim raw_data[15:0]+1$	数据取反
3	当 $raw_addr[20:10]=1024$ 时，rom_addr 取 1023， 其他情况取 $\sim raw_addr[19:10]+1$	$\sim raw_data[15:0]+1$	地址镜像翻转 数据取反

2、主控制器模块 mcu 的设计

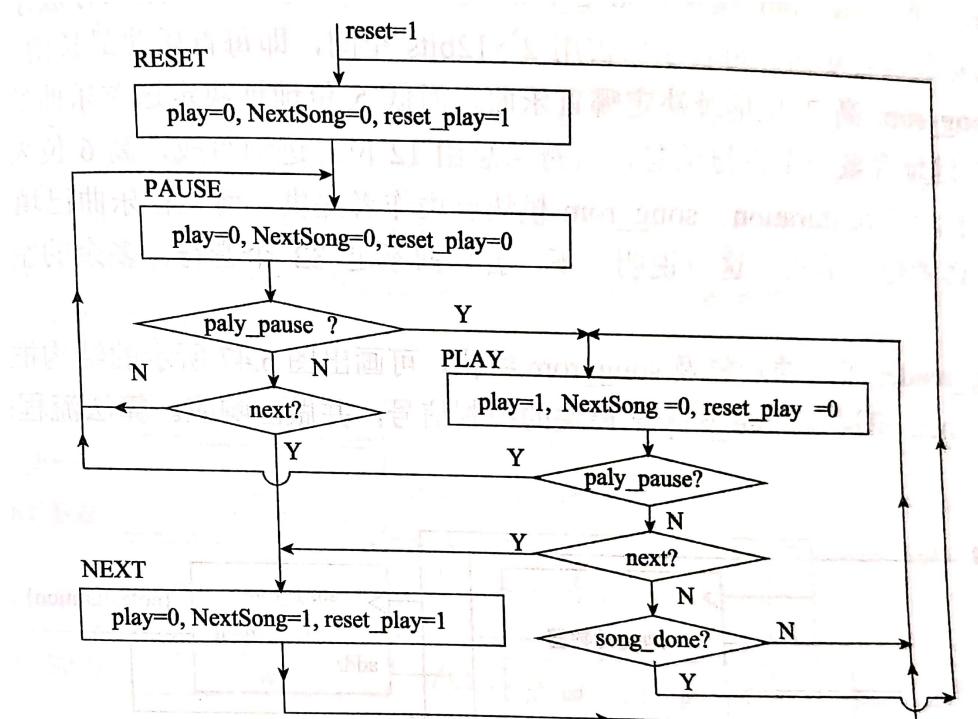
主控制模块 mcu 有响应按键信息、控制系统播放两大任务，下表为其端口定义。

引脚名称	I/O	引脚说明
clk	Input	100MHz 时钟信号
reset	Input	复位信号，高电平有效
play_pause	Input	来自按键处理模块的“播放/暂停”控制信号，一个时钟周期宽度的脉冲
next	Input	来自按键处理模块的“下一曲”控制信号，一个时钟周期宽度的脉冲
play	Output	输出控制信号，高电平表示播放，控制 song_reader 模块是否要播放
reset_play	Output	时钟周期宽度的高电平复位脉冲 reset_play，用于同时复位模块 song_reader 和 note_player
song_done	Input	song_reader 模块的应答信号，一个时钟周期宽度的高电平脉冲，表示一曲播放结束
song[1:0]	Output	当前播放乐曲的序号

根据设计要求，模块 mcu 的原理框图如下所示。图中的 2 位二进制计数器用来计算乐曲序号（song）。



控制器的工作流程图如下所示。



3、乐曲读取模块 song_reader 的设计

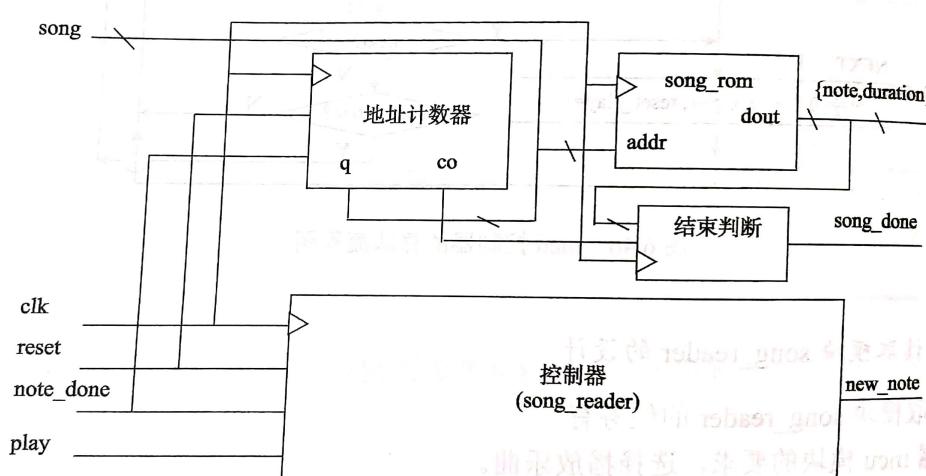
根据 mcu 模块的要求，选择播放乐曲。

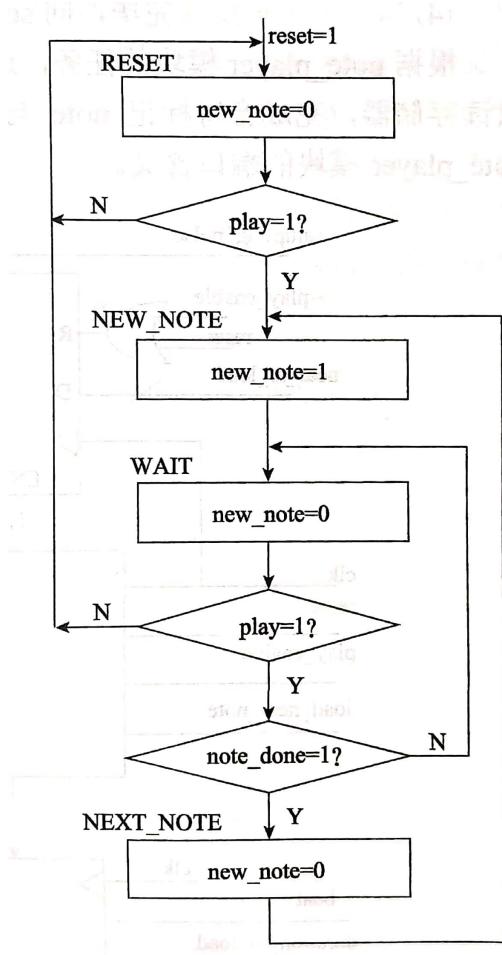
响应 note_player 模块请求，从 song_rom 中逐个取出音符 {note,duration} 送给 note_player 模块播放。

判断乐曲是否播放完毕，若播放完毕，song_reader 模块需包含下表所示的输入、输出端口。

引脚名称	I/O	引脚说明
clk	Input	100MHz 时钟信号
reset	Input	复位信号，高电平有效
play	Input	来自 mcu 的控制信号，高电平要求播放
song[1:0]	Input	来自 mcu 的控制信号，当前播放乐曲的序号
note_done	Input	即模块 note_player 的应答信号，一个时钟周期宽度的脉冲，表示一个音符播放结束并索取新音符
song_done	Output	给 mcu 的应答信号，当乐曲播放结束，输出一个时钟周期宽度的脉冲，表示乐曲播放结束
note[5:0]	Output	音符标记
duration[5:0]	Output	音符的持续时间
new_note	Output	给模块 note_player 的控制信号，一个时钟周期宽度的高电平脉冲，表示新的音符需播放

song_rom 是一个只读存储器，用来存放乐曲，容量为 $2^5 \times 12\text{bits}$ 。共存放四首乐曲，每首乐曲占用 $2^5 \times 12\text{bits}$ 空间，即每首乐曲最长由 32 个音符组成。因此，song_rom 高 2 位地址决定哪首乐曲，而低 5 位地址决定这首乐曲的那个音符。song_rom 每个地址存放一个音符信息，音符信息由 12 位二进制组成，高 6 位表示音符标记 note，低 6 位表示音长 duration。根据 song_reader 模块的功能及 song_rom 结构，可画出结构框图，控制器主要负责接收 mcu 模块与 note_player 模块的控制信号，并作出响应。结构框图和算法流程图如下所示。





4、音符播放模块 note_player 的设计

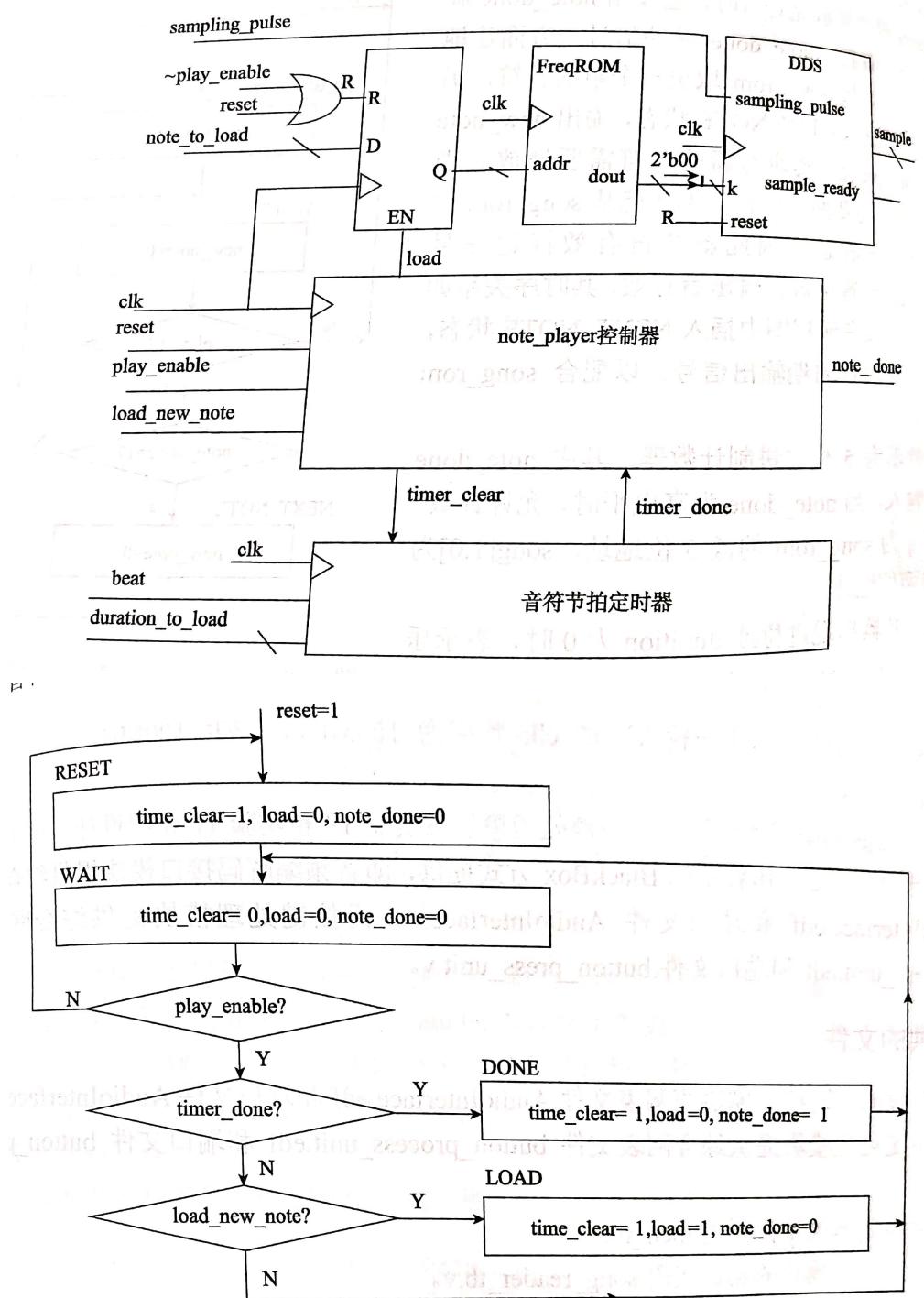
音符播放模块 note_player 是本实验的核心模块，它主要任务有：

- (1) 从 song_reader 模块接收需播放的音符 {note, duration}；
- (2) 根据 note 值找出 DDS 的相位增量 k；
- (3) 以 48kHz 速率从 Sine ROM 取出正弦样品送给音频编解码器接口模块；
- (4) 当一个音符播放完毕，向 song_reader 模块索取新的音符。

根据 note_player 模块的任务，进一步划分功能单元，如图 6.50 所示，图中 FreqROM 为只读存储器，完成音符标记 note 与 DDS 模块的相位增量 k 查找表关系。下表所示为 note_player 模块的端口含义。

引脚名称	I/O	引脚说明
clk	Input	系统时钟信号，外接 sys_clk
reset	Input	复位信号，高电平有效，外接 mcu 模块的 reset_play
play_enable	Input	来自 mcu 模块的 play 信号，高电平表示播放
note_to_load[5:0]	Input	来自 song_reader 模块的音符标记 note，表示需播放的音符
duration_to_load[5:0]	Input	来自 song_reader 模块的音符持续时间 duration，表示需播放音符的音长
load_new_note	Input	来自 song_reader 模块的 new_note 信号，一个时钟周期宽度的高电平脉冲，表示新的音符需播放
note_done	Output	给 song_reader 模块的应答信号，一个时钟周期宽度的高电平脉冲，表示音符播放完毕
sampling_pulse	Input	来自同步化电路模块的 ready 信号，频率 48kHz，一个时钟周期宽度的高电平脉冲，表示索取新的正弦样品
beat	Input	定时基准信号，频率为 48Hz 脉冲，一个时钟周期宽度的高电平脉冲
sample [15:0]	Output	正弦样品输出

note_player 控制器负责与 song_reader 模块接口，读取音符信息，并根据音符信息从 Frequency ROM 中读取相应相位增量 k 送给 DDS 子模块。另外，note_player 控制器还需要控制音符播放时间。note_player 结构框图以及控制器的算法流程如下图所示。



音符定时器为 6 位二进制计数器，beat、timer_clear 分别为使能、清 0 信号，均为高电平有效。定时时间由音长信号 duration_to_load 决定，即 duration_to_load 个 beat 周期，timer_done 为定时结束标志。

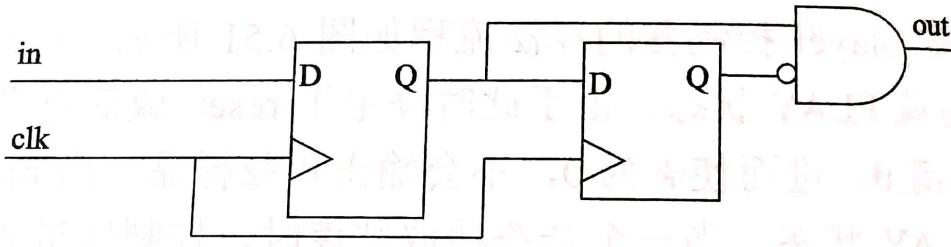
子模块 DDS 的功能就是利用 DDS 技术产生正弦样品。DDS 模块的输入 k 为 22 位二进制，因此需 FreqROM 输出的 20 位相位增量高位加 2 个 0 后接入 DDS。

5、同步化电路

由于音频编解码接口模块和其他模块使用不同的时钟，因此两者之间的控制及应答信号须进行同步化

处理。

音频编解码接口模块的输出信号 NewFrame 的脉冲宽度为一个 audio_clk 时钟周期，需通过同步化处理，产生与 sys_clk 同步且脉冲宽度为一个 sys_clk 时钟周期的信号 ready。电路如下所示，由同步器和脉冲宽度变换电路组成。



6、时钟管理模块（DCM）

IP 内核时钟管理模块的输入时钟 clk 频率为 100MHz，产生 100MHz 的系统时钟和 12.50MHz 的音频时钟。

四、实验设备

- 1、装有 Vivado 和 ModelSim SE 软件的计算机。
- 2、Nexys Video Artix-7 FPGA 多媒体音视频智能互联开发系统。
- 3、有源音箱或耳机。

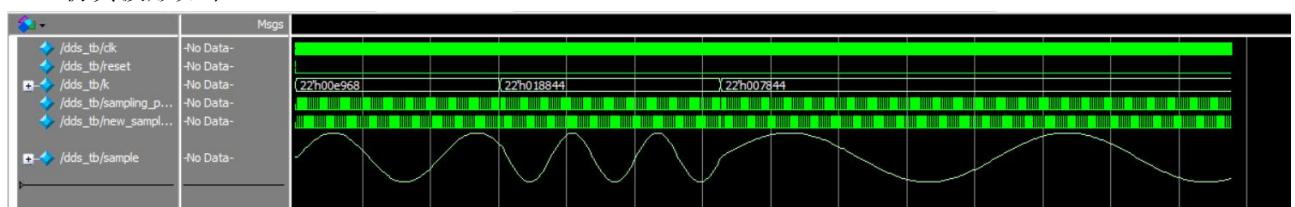
五、实验内容

- 1、编写 DDS 模块的 Verilog HDL 代码，并用 Modelsim 仿真验证。

代码如下：

```
1 module dds(k,clk,reset,sampling_pulse,sample,new_sample_ready);
2   input [21:0]k;//相位增量
3   input clk,reset,sampling_pulse;
4   output [15:0]sample;//正弦信号
5   output new_sample_ready;
6   wire [21:0]raw_addr;//地址处理输入
7   wire [21:0]s;//加法器输出
8   fulladder_n #(.(n(22))addr1(.a(k),.b(raw_addr),.ci(0),.s(s),.co()));
9   dffre #(.(n(22))dffre1(.d(s),.clk(clk),.en(sampling_pulse),.r(reset),.q(raw_addr));
10  wire [9:0]rom_addr;//ROM地址
11  assign rom_addr[9:0] = raw_addr[20]?((raw_addr[20:10]==1024)?1023:(~raw_addr[19:10]+1)):raw_addr[19:10];//地址处理
12  wire [15:0]raw_data;//ROM输出
13  sine_rom sine_rom1(.clk(clk),.addr(rom_addr),.dout(raw_data));
14  wire area;//正弦区域
15  dffre #(.n(1))dffre2(.d(raw_addr[21]),.en(1),.r(0),.clk(clk),.q(area));
16  wire [15:0]data;//正弦样品data
17  assign data[15:0] = area?(~raw_data[15:0]+1):raw_data[15:0];//数据处理
18  dffre #(.(n(16))dffre3(.d(data),.clk(clk),.en(sampling_pulse),.r(0),.q(sample));
19  dffre #(.n(1))dffre4(.d(sampling_pulse),.clk(clk),.en(1),.r(0),.q(new_sample_ready));
20 endmodule
21
22
```

仿真波形如下：



由图中可知，当 k 值较大时，正弦波频率较大，k 值较小时，正弦波的频率较小，k 变化时波的频率随即发生变化，说明 DDS 的设计符合要求。

2、编写 mcu 模块的 Verilog HDL 代码，并用 Modelsim 仿真验证。

代码如下：

```
1 module mcu(clk,reset,play_pause,next,play,reset_play,song_done,song);
2   input clk;//100MHz时钟信号
3   input reset,play_pause,next;
4   input song_done;//一个周期宽度的高电平脉冲，表示乐曲播放结束
5   output play,reset_play;
6   output [1:0]song;//当前播放乐曲的序号
7   wire nextsong;//下一首播放
8   mcu_c mcu_c1(.clk(clk),.reset(reset),.play_pause(play_pause),.next(next),.play(play),.reset_play(reset_play),.song_done(song_done),.nextsong(nextsong));//控制器
9   counter_n #(n(4),.counter_bits(2))counter_n1(.en(nextsong),.r(reset),.clk(clk),.q(song),.co());//二位二进制计数器
0 endmodule
```

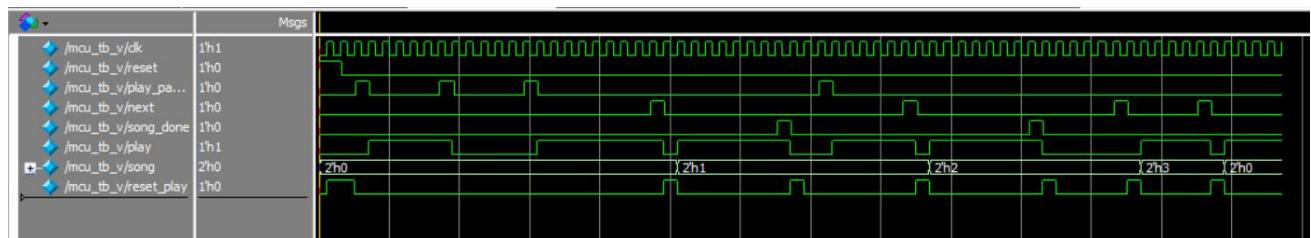
```
module mcu_c(clk,reset,play_pause,next,play,reset_play,song_done,nextsong);
  input clk,reset,play_pause,next,song_done;
  output play,reset_play,nextsong;
  parameter RESET=0,PAUSE=1,PLAY=2,NEXT=3;//状态编码
  reg [1:0]state,nextstate;
  reg play;
  reg reset_play;
  reg nextsong;
  //D寄存器
  always @(posedge clk)begin
    if(reset)
      state=RESET;
    else
      state=nextstate;
  end
  //下一状态和输出电路
  always @(*)begin
    play=0;nextsong=0;reset_play=0;//默认值为0
    case(state)
      RESET:begin
        play=0;
        nextsong=0;
        reset_play=1;
        nextstate=PAUSE;
      end
      PAUSE:begin
        play=0;
        nextsong=0;
        reset_play=0;
        if(play_pause)
          nextstate=PLAY;
        else begin
          if(next)
            nextstate=NEXT;
          else
            nextstate=PAUSE;
        end
      end
    endcase
  end
endmodule
```

```

end
PLAY:begin
    play=1;
    nextsong=0;
    reset_play=0;
    if(play_pause)
        nextstate=PAUSE;
    else begin
        if(next)
            nextstate=NEXT;
        else begin
            if(song_done)
                nextstate=RESET;
            else
                nextstate=PLAY;
        end
    end
end
NEXT:begin
    play=0;
    nextsong=1;
    reset_play=1;
    nextstate=PLAY;
end
default:nextstate=RESET;
endcase
end
endmodule

```

仿真波形如下：



由图中可知，当 Reset=1 时，state=RESET，play = 0，nextsong = 0，reset_play = 1；state=PAUSE，play = 0，nextsong = 0，reset_play = 0；当 reset=0，play_pause=1 时，state=play，play = 1，nextsong = 0，reset_play = 0；当 reset=0，play=1，play_pause=1 时，state=PAUSE，play = 0，nextsong = 0，reset_play = 0；当 play_pause=0，next=1 时，state=NEXT，play = 0，nextsong = 1，reset_play = 1。综上所述，该 mcu 的设计符合题意。

3、编写 song_reader 模块的 Verilog HDL 代码，并用 Modelsim 仿真验证。

代码如下：

```

module song_reader(clk,reset,play,song(note_done,song_done,note,duration,new_note));
    input clk,reset;
    input play;//高电平要求播放
    input note_done;//表示一个音符播放结束并索取新音符
    input [1:0]song;
    output song_done;
    output new_note;//新的音符需播放
    output [5:0]note;//音符标记
    output [5:0]duration;//音符的持续时间
    //控制器
    song_reader_c song_reader_c1(.clk(clk),.reset(reset),.note_done(note_done),.play(play),.new_note(new_note));
    wire [4:0]q;//音符地址后五位
    wire co;//32个音符播放完毕
    //地址计数器
    counter_n #(.(n(32)),.counter_bits(5)) counter_n1(.clk(clk),.r(reset),.en(note_done),.q(q),.co(co));
    //乐曲选择
    song_rom song_rom1(.clk(clk),.addr({song,q}),.dout({note,duration}));
    wire [5:0]duration;
    //结束判断
    over over1(.clk(clk),.co(co),.duration(duration),.song_done(song_done));
endmodule

```

```

module song_reader_c(clk,reset,play,note_done,new_note);
    input clk,reset,play,note_done;
    output new_note;
    parameter RESET=0,NEW_NOTE=1,WAIT=2,NEXT_NOTE=3;//状态编码
    reg [1:0]state,nextstate;
    reg new_note;
    //D寄存器
    always @(posedge clk)begin
        if(reset)
            state=RESET;
        else
            state=nextstate;
    end
    //下一状态和输出电路
    always @(*)begin
        new_note=0;
        case(state)
            RESET:begin
                new_note=0;//默认值为0
                if(play)
                    nextstate=NEW_NOTE;
                else
                    nextstate=RESET;
            end
            NEW_NOTE:begin
                new_note=1;
                nextstate=WAIT;
            end
        endcase
    end
endmodule

```

```

WAIT:begin
    new_note=0;
    if(play)
        if(note_done)
            nextstate=NEXT_NOTE;
        else
            nextstate=WAIT;
    else
        nextstate=RESET;
end
NEXT_NOTE:begin
    new_note=0;
    nextstate=NEW_NOTE;
end
default:nextstate=RESET;
endcase
end

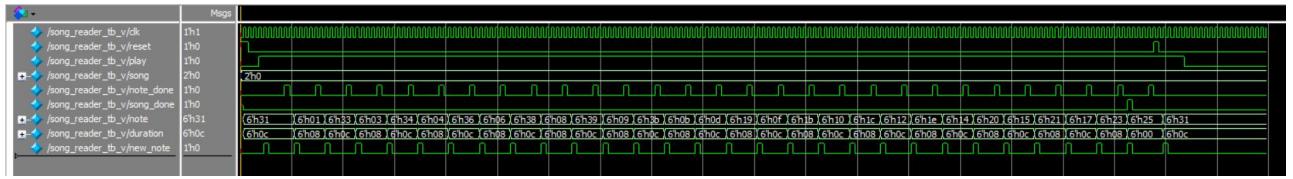
```

```

module over(co,clk,duration,song_done);
    input co,clk;
    input [5:0]duration;
    output song_done;
    wire q;
    assign song_done=((co==1)|| (duration==0))&&~q;//通过或门输出一个时钟周期的信号
    //d触发器
    dffre #(.n(1)) d1(.clk(clk),.en(1),.r(0),.d((co==1)|| (duration==0)),.q(q));
endmodule//脉宽变换电路

```

仿真波形如下：



由波形可知，当 reset=1 时，state=RESET，new_note=0；当 play=1 时，state=NEW_NOTE，new_note=1；state=WAIT，new_note=0；当 play=1，note_done=1 时，state=NEXT_NOTE，new_note=0；然后 state=NEW_NOTE，new_note=1，即读取下一个音符信息。综上所述，该 song_reader 设计符合题意。

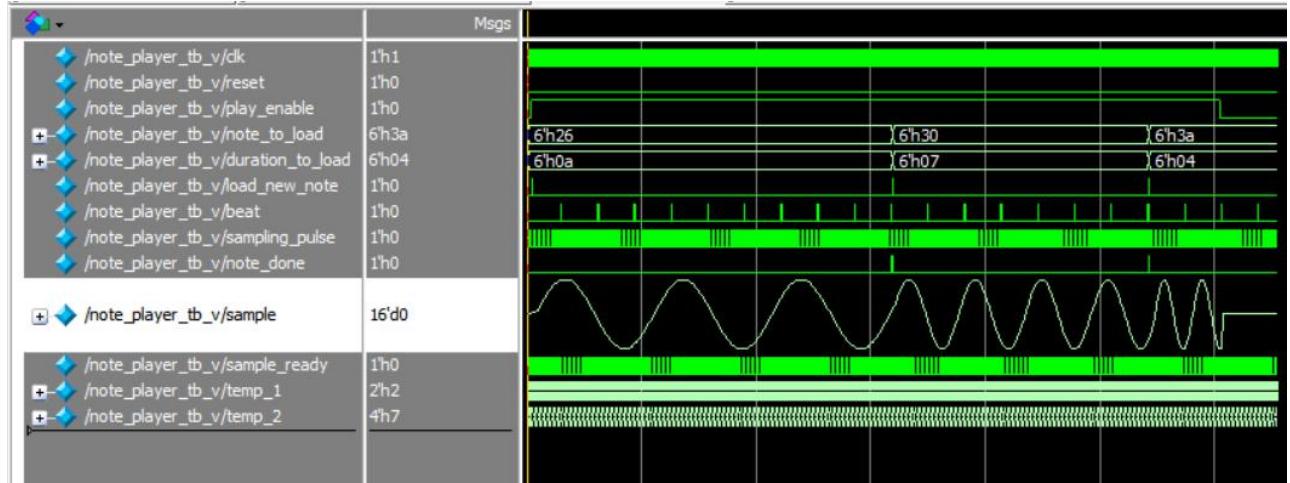
4、编写 note_reader 模块的 Verilog HDL 代码，并用 Modelsim 仿真验证。

代码如下：

```
module note_player(clk,reset,play_enable,note_to_load,duration_to_load,load_new_note,note_done,sampling_pulse,beat,sample,sample_ready);
input clk;//系统时钟信号，外接sys_clk
input reset,play_enable,load_new_note,sampling_pulse;
input beat;//定时基准信号，频率为48HZ脉冲，一个时钟周期宽度的高电平脉冲
input [5:0]note_to_load;
input [5:0]duration_to_load;
output note_done,sample_ready;
output [15:0]sample;
wire r;
assign r=~play_enable||reset;
wire timer_clear;//清零信号
wire timer_done;//定时结束标志
//控制器
note_player_c note_player_c1(.clk(clk),.reset(reset),.play_enable(play_enable),.load_new_note(load_new_note),.note_done(note_done),
.|.timer_clear(timer_clear),.timer_done(timer_done),.load(load));
wire [5:0]q;
//d触发器
dffre #(n(6)) d1(.r(r),.en(load),.d(note_to_load),.clk(clk),.q(q));
wire [19:0]dout;
//FreqROM
frequency_rom frequency_rom1(.clk(clk),.addr(q),.dout(dout));
//DDS
dds dds1(.clk(clk),.sampling_pulse(sampling_pulse),.k({2'b00,dout}),.reset(r),.sample(sample),.new_sample_ready(sample_ready));
//音符节拍定时器
timer #(counter_bits(6)) timer1(.clk(clk),.r(timer_clear),.en(beat),.q(duration_to_load),.done(timer_done));
endmodule
```

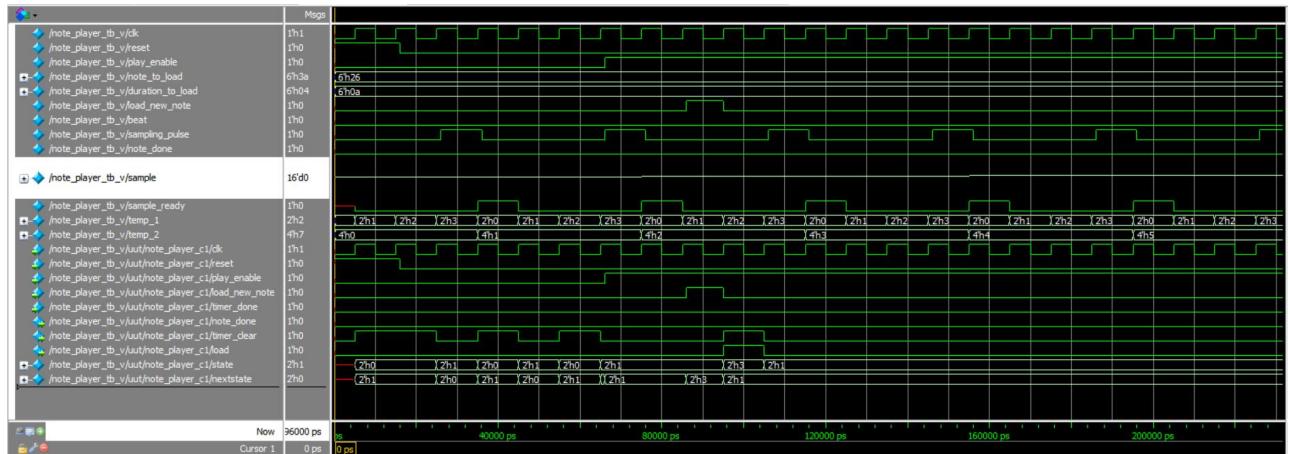
```
module note_player_c(clk,reset,play_enable,load_new_note,timer_clear,timer_done,note_done,load);
input clk,reset,play_enable,load_new_note,timer_clear,timer_done;
output note_done,timer_clear,load;
parameter RESET=0,WAIT=1,DONE=2,LOAD=3;//状态编码
reg [1:0]state,nextstate;
reg timer_clear;
reg load;
reg note_done;
//D寄存器
always @(posedge clk)begin
    if(reset)
        state=RESET;
    else
        state=nextstate;
end
//下一状态和输出电路
always @(*)begin
    timer_clear=0;load=0;note_done=0;//默认值为0
    case(state)
        RESET:begin
            timer_clear=1;
            load=0;
            note_done=0;
            nextstate=WAIT;
        end
        WAIT:begin
            timer_clear=0;
            load=0;
            note_done=0;
            if(play_enable)begin
                if(timer_done)begin
                    nextstate=DONE;
                end
                else
                begin
                    if(load_new_note)
                        nextstate=LOAD;
                    else
                        nextstate=WAIT;
                end
            end
            else
                nextstate=RESET;
        end
        DONE:begin
            timer_clear=1;
            load=0;
            note_done=1;
            nextstate=WAIT;
        end
        LOAD:begin
            timer_clear=1;
            load=1;
            note_done=0;
            nextstate=WAIT;
        end
    endcase
end
endmodule
```

仿真波形如下：



由波形可知，仿真所得结果与书上老师提供的参考波形一致，且当 note_to_load 值越大，正弦波频率越大，当 play_enable=0 时，输出为 0。

由以下局部放大图可知：



当 reset=1 时，state=RESET, time_clear=1, load=0, note_clear=0; 然后进入 state=WAIT, time_clear=1, load=0, note_clear=0; 当 play_enable=1, timer_done=0, load_new_note=1 时，state=LOAD, time_clear=1, load=1, note_clear=0。综上所述，该 note_player 符合题意。

5、编写 1000 分频器模块的 Verilog HDL 代码及其测试代码，并用 Modelsim 仿真验证。

由于在 lab7 已经写过分频器的对应代码，因此此处的 1000 分频器使用的是 lab7 的代码。代码如下：

```

module counter_n(r,en,co,q,clk);
parameter n = 2;
parameter counter_bits=1;
input r,en,clk;
output co;
output [counter_bits-1:0]q;
reg [counter_bits-1:0]q=0;
assign co=(q==(n-1))&& en;//进位输出
always @ (posedge clk)begin
    if(r==1)
        q <= 0;//复位
    else if(en)
    begin
        if(q==(n-1))
            q=0;//同步清零
        else
            q=q+1;
    end
end
endmodule

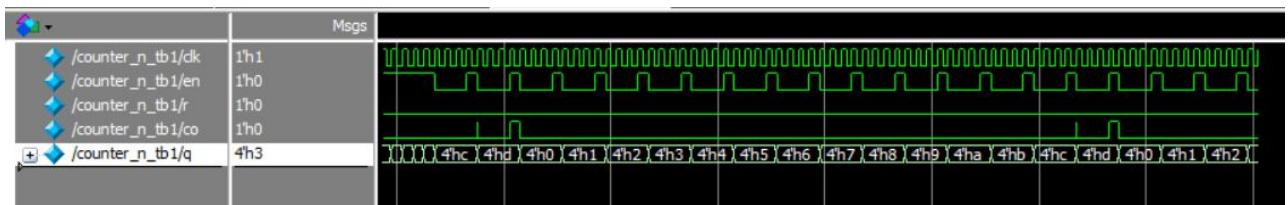
```

```

`timescale 1ns/1ps
module counter_n_tb1;
reg clk;
reg en;
reg r;
wire co;
wire [3:0]q;
counter_n #(n(14),.counter_bits(4))i1(.clk(clk),.r(r),.en(en),.q(q),.co(co));
always #50 clk=~clk;
initial
begin
    clk = 0;
    r = 0;
    en = 0;
    #(51)r=1;
    #(100)r=0;en=1;
    #(800)
    repeat(20)begin
        #(100*3)en=1;
        # 100 en=0;
    end
    # 100 $stop;
end
endmodule

```

仿真波形如下：



由波形可知，由于测试代码中是一个 14 进制的计数器，与波形所得结果一致，因此该分频器符合题意。

6、编写同步化电路模块的 Verilog HDL 代码及其测试代码，并用 Modelsim 仿真验证。

代码如下：

```
module tongbu(in,clk,out);
    input in,clk;
    output out;
    wire q0,q1;
    assign out = q0&&~q1;//与门实现输出一个时钟周期长度的信号
    dffre #(n(1)) d1(.d(in),.q(q0),.en(1),.r(0),.clk(clk));
    dffre #(n(1)) d2(.d(q0),.clk(clk),.en(1),.r(0),.q(q1));
endmodule//同步化电路
```

```
`timescale 1ns/1ps
module tongbu_tb;
    reg clk;
    reg in;
    wire out;
    tongbu tongbu1(.clk(clk),.in(in),.out(out));
    //clock
    always #50 clk=~clk;
    initial
    begin
        // Initialize Inputs
        clk = 0;
        in = 0;

        #(51) in=0;
        #(50) in=1;
        #(50) in=0;
        #(200)
        repeat(10)begin
            #(50*3) in=1;
            #(50) in=0;
        end
        #(100) $stop;
    end
endmodule
```

仿真波形如下：



由图中波形可知，输出的信号均符合一个时钟周期，因此该同步化电路设计符合题意。

7、编写次顶层 music_player 模块的 Verilog HDL 代码，并用 Modelsim 仿真验证。

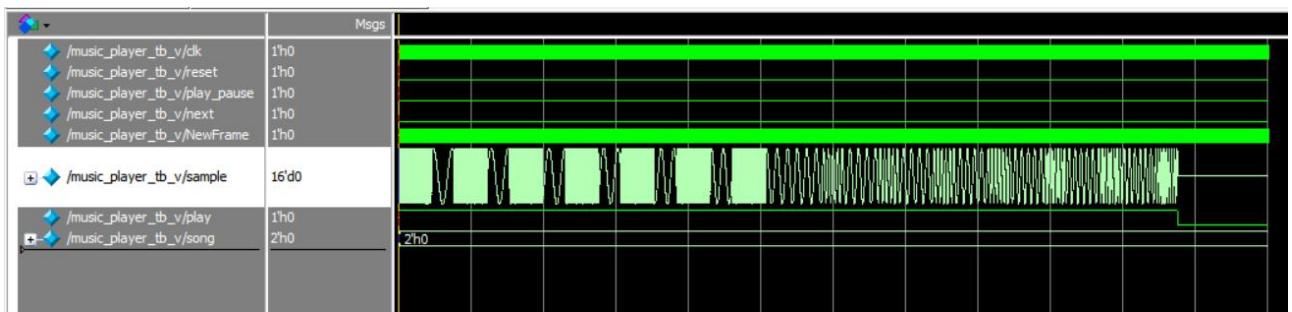
代码如下：

```

module music_player #(parameter sim = 0)(clk,reset,play_pause,next,NewFrame,sample,play,song);
input clk,reset,play_pause,next,NewFrame;
output [15:0]sample;
output play;//高电平表示播放
output [1:0]song;
wire song_done;
wire reset_play;
wire play;
mcu mcu1(.clk(clk),.next(next),.play_pause(play_pause),.reset(reset),.song(song),.play(play),
.reset_play(reset_play),.song_done(song_done));//主控制器
wire [5:0]duration;
wire [5:0]note;
wire new_note;
wire note_done;
//乐曲读取
song_reader song_reader1(.song(song),.play(play),.reset(reset_play),.note_done(note_done),.clk(clk),
.song_done(song_done),.duration(duration),.note(note),.new_note(new_note));
wire beat;
wire ready;
//音符播放
note_player note_player1(.clk(clk),.reset(reset_play),.beat(beat),.play_enable(play),.note_to_load(note),
.duration_to_load(duration),.load_new_note(new_note),.note_done(note_done),.sample(sample),.sampling_pulse(ready),.sample_ready());
//同步化电路
tongbu tongbu1(.clk(clk),.in(NewFrame),.out(ready));
//节拍基准产生器
counter_n #(n(sim?64:1000),.counter_bits(sim?6:10))counter_n1(.clk(clk),.en(ready),.r(0),.q(),.co(beat));
endmodule

```

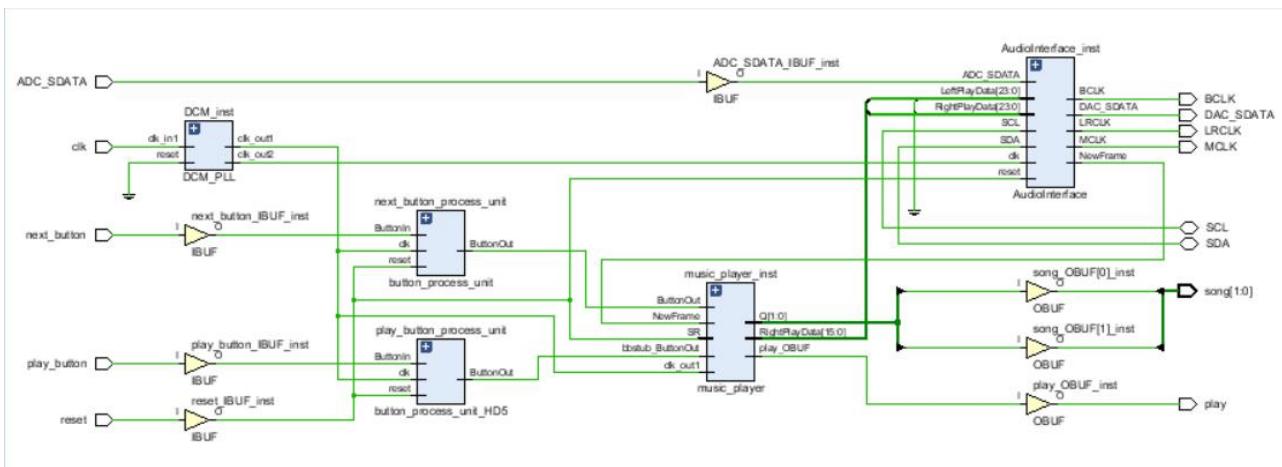
仿真波形如下：



由波形可知，仿真所得结果与书上老师提供的参考波形一致，因此 music_player 设计符合题意。

8、Vivado 综合

综合所得电路图如下所示：



左边的 clk、next_button、play_pause_button、reset 等为输入，其中 clk 由 DCM 内核生成的时钟控制，next_button、play_pause_button、reset 信号由按键输入，右边 song、play 等是输出，表示为乐曲的播放暂停以及播放的是哪一首乐曲。

演示中，按下 play_pause_button 按键，乐曲开始播放，按下 next_button 按键，乐曲切换到下一首，

按下 play_pause_button 按键，乐曲暂停，按下 reset 按键，系统复位。与预期结果相符合。

六、思考题

1、因为 reset 是异步置零信号，当 reset=1 时，系统置零，reset=0 时，reset 信号对系统无影响，因此如果 reset 信号存在颤动，在 reset=1 后产生的颤动对系统都没有影响。而 next_button、play_pause_button 按键如果有颤动将对系统产生影响比如在 next=1 后产生颤动，又出现了 next=1 的情况，那么乐曲将会从第一首跳到第三首，与预期结果不一致；同样如果 play_pause=1 时，play_pause_button 存在颤动，则可能出现乐曲暂停后又播放的情况，与预期结果不一致，因此需要对 next_button、play_pause_button 按键进行防颤动和同步化处理。

2、存在。在 RESET 到 PAUSE 和 NEXT 到 PLAY 的状态转换过程中没有进行 next 和 play_pause 的判断，因此在这两个转换过程中接收不到按键信息，即按下按键不会改变系统状态。不用修改设计，因为这两个状态转换只有很短的一个时钟周期的时间。

七、总结与体会

这次的实验相比之前的 lab5 和 lab7 难度有了很大的提升，特别是刚开始做 DDS 的设计，完全理解书上的系统框图和地址数据处理方法花费了很多时间，导致一开始的进度较慢，到后来完成了 DDS 的仿真进入 mcu 模块的设计之后，由于之前没有做过控制器的设计，又回去学习控制器的设计方法，导致也花费了较多时间。不过由于做过 lab7，因此对时序电路的设计比较熟悉，在学习了控制器的设计方法之后对模块的设计也逐渐熟练，因此之后写代码的速度提升了很多。在仿真的过程中也遇到了较多的问题，经常会因为一个小错误浪费很多时间，比如计数器中 n 的位数设置错误，因此在写代码的时候要注意检查每一句代码是否正确，避免在仿真时浪费过多的时间。总的来说，通过本次实验，我对控制器的设计方法有了更多的了解，对系统“自顶而下”的数字系统设计方法有了更好的掌握，我意识到数字系统设计是一个“牵一发而动全身”的过程，其中一个模块里面的一个小错误可能会导致整个系统出现致命的错误，而且我们在设计过程中都有教材的辅助和老师的帮助，因此我们更应该努力学习专业知识，不断提高自己的综合能力。