



华中科技大学

接口技术综合实验报告

姓 名： 洪炜豪
学 号： U202115512
小组成员： 金亚璞
学 院： 计算机科学与技术学院
专 业： 计算机科学与技术
班 级： CS 2106
指导教师： 胡迪青

2023 年 11 月 20 日

目录

1.实验目的	1
2.实验环境	1
3.项目设计与实现	2
3.1 总体设计与实现	2
3.2 硬件实现	2
3.3 软件实现	6
4.系统运行结果	7
5.总结	9
6.源代码	10

1. 实验目的

基于 Imagination 大学计划开源的 RVfpgaSoC，通过讲课和实验，使学生了解和掌握接口技术，并采用基本的接口模块，运用 EDA 工具通过 IP 集成的方式搭建一个简单的嵌入式应用系统。

目标 1：熟悉硬件设计工具的功能、特点及使用方法，掌握基于主流设计工具设计硬件模块或系统的流程和方法，使学生具备硬件系统的开发和调试能力；

目标 2：基于接口技术理论知识，根据实验任务要求，设计硬件功能部件和简单系统，掌握硬件及系统设计方法，培养学生发现问题、分析问题、解决问题的实践能力；

目标 3：通过实验检查和验收过程中的问答，阐述实验设计等活动，培养学生与专业有关的沟通与表达能力。

实验要求：在前面实验的基础上，在 RVfpga_SoC 处理器系统中至少再添加一个新的外设模块，完成系统的硬件设计，生成比特流。

注：添加的新外设必须是前面实验没有使用过的，这个外设模块可以是 Vivado 自带的、也可以是现成的免费的 IP 模块、还可以是自定义的 IP 模块。

在上述实现的 RVfpga_SoC 处理器系统上，编写一个综合应用软件。该综合应用系统要能够充分展示同学们的想象力和创造性。

2. 实验环境

Vivado:

v2019.2 (64-bit)

SW Build: 2708876 on Wed Nov 6 21:40:23 MST 2019

IP Build: 2700528 on Thu Nov 7 00:09:20 MST 2019

VSCode:

版本: 1.84.2 (user setup)

提交: 1a5daa3a0231a0fbba4f14db7ec463cf99d7768e

日期: 2023-11-09T10:51:52.184Z

OS: Windows_NT x64 10.0.19044

处理器:

11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz

操作系统:

版本 Windows 10 专业版

版本号 21H2

安装日期 2021/8/5

操作系统内部版本 19044.3086

3.项目设计与实现

3.1 总体设计与实现

先设计 pwm 音频输出外设模块，集成内置音频文件，然后将 PWM 模块集成到 RVfpga_SoC 处理器系统中。完成硬件设计后，使用 Vivado 生成比特流文件。接着在 platform 上编写 c 语言代码运行一个系统，这个系统是时钟系统，支持设定日期时间，闹铃，倒计时，时间到就激活 pwm 模块响铃。最后进行测试。

3.2 硬件实现

1. 设计 PWM 音频输出功能：

- 设计 PWM 模块以实现音频输出。
- 集成内置音频文件。

PWM（Pulse Width Modulation）最简单的理解就是 MCU 内部有个定时器，定时器特定的时间内将 GPIO 的电平翻转一下。对于翻转的时间间隔的比例就是占空比，而持续这两个过程则可以算为一个周期。

PWM 模块被设计用来生成对应于音乐音符的音频音调，我的设计中简单地演奏了《See You Again》的前奏。模块设置为能够产生不同频率的方波，这些频率对应于音乐音符。下面是代码的详细解释：

模块和宏定义

```
```verilog
```

```
`timescale 1ns / 1ps
```

```
```
```

这行代码定义了仿真中使用的时间单位和精度。1ns 表示时间单位是纳秒，1ps 表示时间精度是皮秒。

```
```verilog
```

```
// 高低中音宏定义
```

```
`define L1 262 //低音 1
```

```
...
```

```
`define H7 1976 //高音 7
```

```
```
```

这些行定义了音符的频率值，低音（L）、中音（C）、高音（H），用于后续 PWM 信号的产生。

模块接口

```
```verilog
```

```
module My_Controller(
```

```
 input clk,
```

```
 input rst,
```

```
 input [31:0] enable,
```

```

 output reg AUD_PWM,
 output wire AUD_SD
);
 ...

```

- `clk` 是模块的时钟输入。
- `rst` 是复位信号，用于重置模块状态。
- `enable` 是一个 32 位的输入信号，用于启用或禁用音频输出。
- `AUD\_PWM` 是音频输出的 PWM 信号。
- `AUD\_SD` 是一个输出信号，当 `enable` 为高时，它允许音频输出。

```

状态计数器
```verilog
localparam state_top=24'd12500000-1;
reg [23:0] state_cnt;
```

```

这里定义了一个局部参数 `state\_top` 和一个状态计数器 `state\_cnt`。每当 `state\_cnt` 达到 `state\_top`（假定时钟频率下的 250ms 计数值）时，状态计数器复位。

```

状态计数器逻辑
```verilog
always@(posedge clk or negedge rst)
    if(!rst)
        state_cnt<=0;
    else if(state_cnt<state_top)
        state_cnt<=state_cnt+1;
    else
        state_cnt<=0;
```

```

这个始终块控制状态计数器的行为。在每个时钟上升沿，如果复位信号为低，则计数器清零；如果计数器值小于 `state\_top`，则计数器增加；否则，计数器复位。

```

状态机逻辑
```verilog
reg [20:0] cnt_top;
wire state_cnt_done=(state_cnt==state_top)?1:0;
always@(posedge clk or negedge rst)
    if(!rst)
        state<=0;
    else if(state_cnt_done)begin
        if(state<73)
            state<=state+1;
        else

```

```

        state<=0;
    end
    else
        state<=state;
    ...

```

这部分定义了一个 7 位的状态寄存器 `state`，且在每个时钟上升沿更新状态。如果复位信号为低，状态清零；如果状态计数器完成（`state_cnt_done`），则状态递增或复位。

```

#### 音符选择逻辑
```verilog
always@(*)
begin
 case(state)
 0 :cnt_top<=`C5;
 ...
 73 :cnt_top<=`C5;
 default;;
 endcase
end
...

```

这个组合逻辑块根据当前状态选择对应的音符频率值。每种状态对应一个 `case` 分支，用于设置 `cnt\_top`，这决定了 PWM 信号的频率。

```

PWM 频率计数器
```verilog
reg [26:0] cnt;
always@(posedge clk or negedge rst)
    if(!rst)
        cnt<=0;
    else if(cnt<50_000_000/cnt_top-1)
        cnt<=cnt+1;
    else
        cnt<=0;
...

```

定义了一个计数器 `cnt`，用于产生 PWM 信号。

```

#### PWM 产生逻辑
```verilog
always@(posedge clk or negedge rst)
 if(!rst)
 AUD_PWM<=0;
 else
 AUD_PWM<=(cnt<cnt_top/2)?1:0;

```

...

这个始终块生成 PWM 信号。它根据 `cnt\_top` 控制 PWM 信号的占空比，`cnt` 在到达 `cnt\_top/2` 时翻转 `AUD\_PWM` 信号的状态。

#### ### 主要功能

主要实现了一个有限状态机，每个状态对应旋律中的一个音符。`state\_cnt` 计数器为状态转换之间的延迟，以保持每个音符的持续时间。当 `state\_cnt\_done` 信号被断言时，状态切换，改变正在播放的音符。

`cnt\_top` 值是当前音符频率的周期，用于生成 PWM 信号。`AUD\_PWM` 信号以音符的频率切换，当连接到扬声器或音频输出设备时，创建可听见的音调。

`AUD\_SD` 信号用于根据 `enable` 输入启用或禁用音频输出，允许控制音频活动的时间。

总体来说，这个模块通过控制 `AUD\_PWM` 信号的占空比，产生一系列对应不同音符频率的方波信号，用于在连接的扬声器上播放音乐。

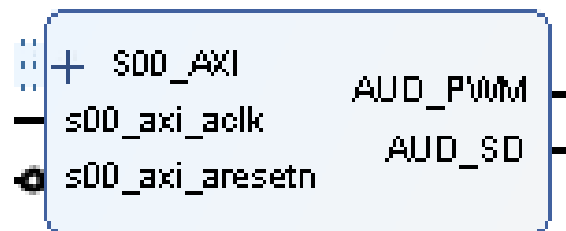


图 3-1 封装好后的 pwm 模块

## 2. 选择和集成 PWM 音频输出模块：

- 在 Vivado 中选择自定义的 IP 模块 PWM 模块。
- 将 PWM 模块集成到 RVfpga\_SoC 处理器系统中。在处理器与 PWM 模块之间建立适当的接口，并确保它们能够有效地通信。

axi2wb_intcon_wrapper_0						
o_ram_axi4 (32 address bits : 4G)						
ram	ram	Reg	0x0000_0000	128M		0x07FF_FFFF
o_user_axi4 (32 address bits : 4G)						
PWM_w_Int_0	S00_AXI	S00_AXI_reg	0x8012_0000	64K		0x8012_FFFF
axi_gpio_0	S_AXI	Reg	0x8010_0000	64K		0x8010_FFFF
axi_iic_0	S_AXI	Reg	0x8013_0000	64K		0x8013_FFFF
My_PWM_0	S00_AXI	S00_AXI_reg	0x8014_0000	64K		0x8014_FFFF
swerv_wrapper_verilog_0						
ifu_axi (32 address bits : 4G)						
axi2wb_intcon_wrapper_0	i_ifu_axi4	reg0	0x0000_0000	4G		0xFFFF_FFFF
lsu_axi (32 address bits : 4G)						
axi2wb_intcon_wrapper_0	i_lsu_axi4	reg0	0x0000_0000	4G		0xFFFF_FFFF
sb_axi (32 address bits : 4G)						
axi2wb_intcon_wrapper_0	i_sb_axi4	reg0	0x0000_0000	4G		0xFFFF_FFFF

图 3-2 地址分配

```
##Video
```

```
set_property -dict { PACKAGE_PIN A11 IOSTANDARD LVCMOS33 } [get_ports { AUD_PWM }];
set_property -dict { PACKAGE_PIN D12 IOSTANDARD LVCMOS33 } [get_ports { AUD_SD }];
```

图 3-3 引脚约束

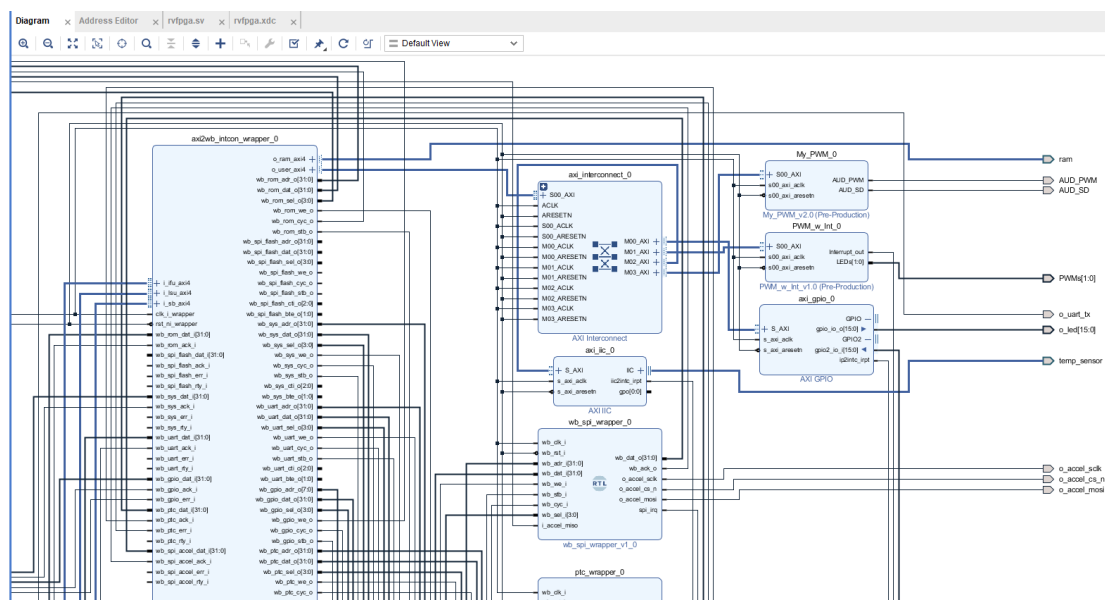


图 3-4 块设计

### 3. 生成比特流:

- 完成硬件设计后，使用 Vivado 生成比特流文件。

## 3.3 软件实现

### 1. 编写时钟系统软件:

- 使用 C 语言在 RVfpga\_SoC 处理器系统上开发时钟系统的软件部分。
- 实现用户界面，允许用户设置日期、时间、响铃时间和倒计时。
- 设计软件以侦听时钟信号，并在指定的时间激活 PWM 模块播放预设的音频。

### 2. 系统功能:

- 日期和时间设置：允许用户设置当前日期和时间。
- 闹铃功能：用户可以设置闹铃时间。
- 倒计时功能：用户可以设置倒计时，到时间后触发闹铃。
- 音频播放：当时钟响起或倒计时结束时，通过 PWM 模块播放音频。

### 3. 测试和调试:

- 在 RVfpga\_SoC 系统上部署软件，并进行测试以确保所有功能按预期工作。
- 调试可能出现的问题，如音频播放质量、定时准确性等。



## 4.系统运行结果

我的时钟系统通过读取 Switch 的输入进行设置，共分配了 5 个操控的开关，分别是 16,4,3,2,1 号 Switch，完成以下几个部分的功能：

16 号开关：进行日期设定，开启后第 15-10 号配置日期，9-6 号配置月份，5-1 号配置日期，支持 2000-1-1 到 2063-12-31 日期设定。

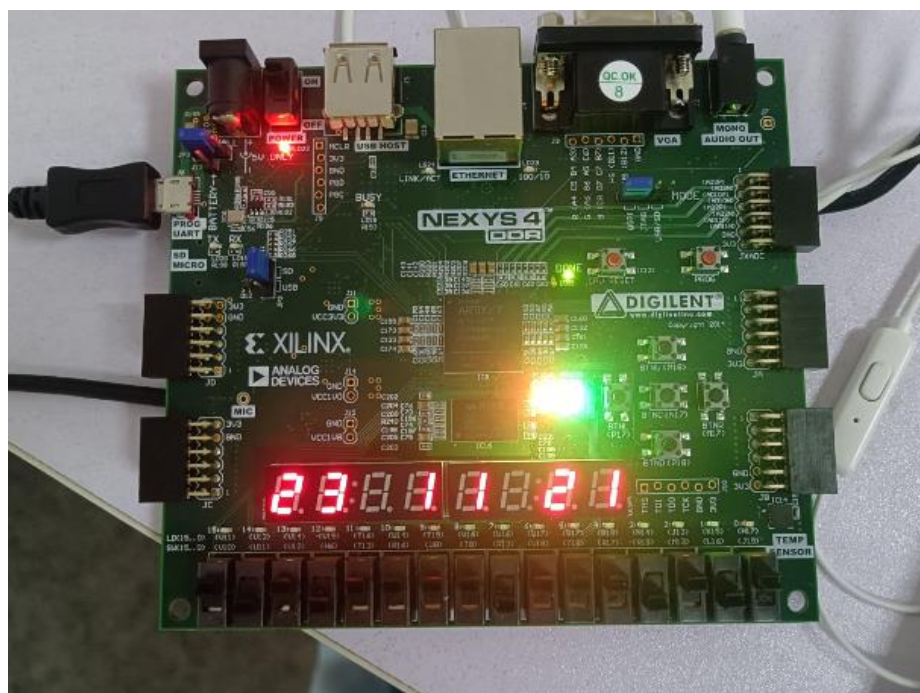


图 4-1 日期配置测试

4 号开关：进行日期的查看，打开后查看当前日期。

3 号开关：打开后进行定时器设定，可设定倒计时 xx 分 xx 秒，时间到了出发闹铃。

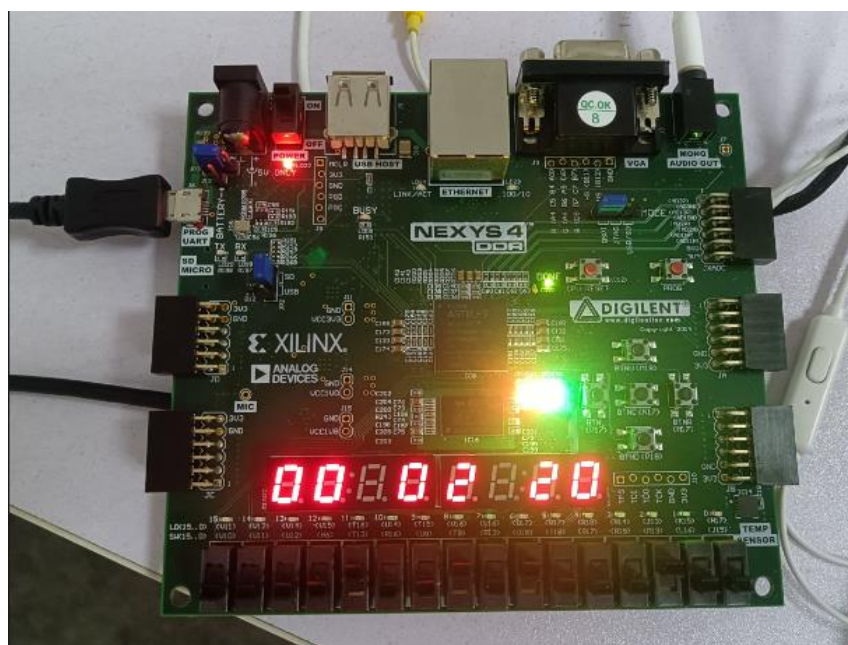


图 4-2 配置定时器

- 3号+4号开关：启动定时器。
- 2号开关：设定闹铃，当系统时间等于闹铃时间时响铃。
- 2号+4号开关：启动闹铃。
- 1号开关：设定当前时间 xx 时 xx 分。

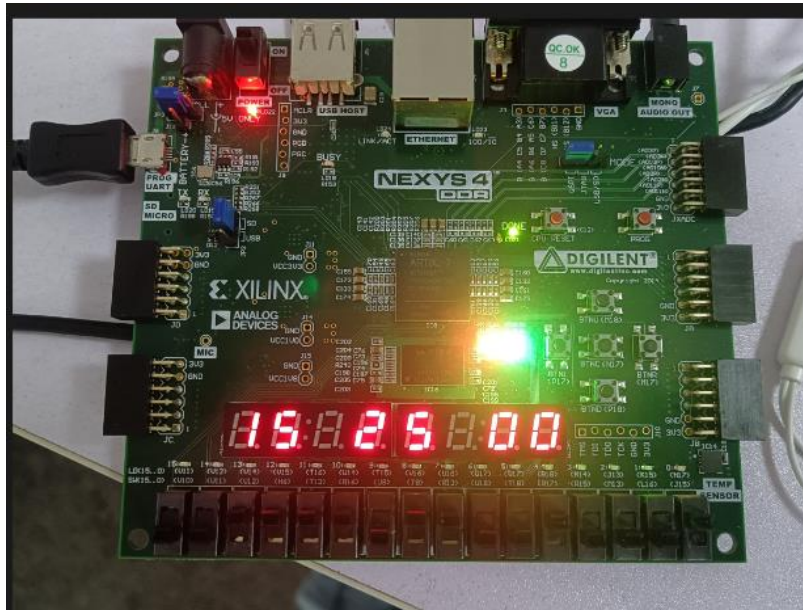


图 4-3 设定当前时间

所有开关闭合时显示当前时间 xx 时 xx 分 xx 秒。

支持日期的自动转换，考虑了闰年，各个月份天数不同的情况。经测试均正常。

### 问题发现与解决:

1. 发现在设定日期时间的时候系统时钟也应该继续运行而不是就此暂停，于是在进入其他 case 时将 h, m, s 保存到 hh, mm, ss 中，在 case 语句结束后，如果发现 hh, mm, ss 改变后，就进行自增，返回 case0 时再将其读到 h, m, s 中进行自增，与此同时把 hh, mm, ss 置为默认值 0xff。
2. 发现 pwm 模块不发出声音：因为人耳只能听到特定频率的声音，而我没有正确设定声音的频率。正确配置频率后解决了问题。

## 5.总结

### 实验体验

- **探索与学习：**通过这个实验，我得以深入探索 RVfpga\_SoC 处理器系统和硬件设计工具（如 Vivado）。这个过程不仅丰富了我的理论知识，还提高了我实践操作的能力。
- **创意实现：**设计时钟系统允许我将创造性思维应用于实际项目中。从设计到实现的过程，让我体验到了从无到有创建一个完整系统的成就感。

### 技术技能的提升

- **硬件设计：**通过在 RVfpga\_SoC 处理器系统中添加 PWM 音频输出模块，我学习了如何设计和集成新的硬件模块。
- **软件开发：**编写时钟系统的软件部分提升了我的编程技能，特别是在嵌入式系统环境中编写和调试代码的能力。

### 实际问题解决

- **调试与优化：**在实验过程中，我遇到了各种技术挑战，如硬件与软件的兼容性问题，以及信号没有正常输出，模块设计不理想等问题。这些问题的解决过程锻炼了我的问题分析和解决问题的能力。
- **创新思维：**设计一个时钟系统需要创意，特别是在功能和用户交互方面。这激发了我的创新思维，让我学会了如何在技术项目中融入创意元素。

### 沟通与表达

- **技术沟通：**通过与同伴和教师的交流，我学会了如何清晰地表达技术观点和设计思路。这种沟通对于理解复杂概念和解决实际问题非常重要。并且我学会了在遇到问题时如何自主寻找解决方案，这增强了我的独立思考和问题解决能力。
- **表达与报告：**撰写实验报告和展示设计过程是这个实验的重要部分。这不仅让我学会了如何书面表达技术思想，还锻炼了我的口头表达能力。

总体来说，这个实验不仅提升了我的技术能力，还锻炼了我的创新思维和解决问题的能力。通过实际设计和实现一个综合应用系统，我得到了宝贵的实践经验，这将对我未来的学习和职业生涯产生深远影响。

最后，再次感谢胡迪青老师和李丹阳助教在实验过程中对我的点拨和帮助！

## 6.源代码

### My\_Controller.v

```
`timescale 1ns / 1ps
///
// Company:
// Engineer:
//
// Create Date: 2023/11/10 18:05:11
// Design Name:
// Module Name: My_Controller
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///
//高低中音宏定义
`define L1 262 //低音 1
`define L2 294 //低音 2
`define L3 330 //低音 3
`define L4 349 //低音 4
`define L5 392 //低音 5
`define L6 440 //低音 6
`define L7 494 //低音 7

`define C1 523 //中音 1
`define C2 587 //中音 2
`define C3 659 //中音 3
`define C4 699 //中音 4
`define C5 784 //中音 5
`define C6 880 //中音 6
`define C7 988 //中音 7

`define H1 1046 //高音 1
`define H2 1175 //高音 2
`define H3 1319 //高音 3
```

```

`define H4 1397 //高音 4
`define H5 1568 //高音 5
`define H6 1760 //高音 6
`define H7 1976 //高音 7

module My_Controller(
 input clk,
 input rst,
 input [31:0] enable,
 output reg AUD_PWM,
 output wire AUD_SD
);
//250ms 跳转一次,需要计数 12500000
localparam state_top=24'd12500000-1;
reg [23:0] state_cnt;
assign AUD_SD = enable[0];
always@(posedge clk or negedge rst)
 if(!rst)
 state_cnt<=0;
 else if(state_cnt<state_top)
 state_cnt<=state_cnt+1;
 else
 state_cnt<=0;
//状态跳转标志信号
wire state_cnt_done=(state_cnt==state_top)?1:0;

reg [4:0] state;
reg [20:0] cnt_top;
always@(posedge clk or negedge rst)
 if(!rst)
 state<=0;
 else if(state_cnt_done)begin
 if(state<31)
 state<=state+1;
 else
 state<=0;
 end
 else
 state<=state;

always@(*)
 begin

```

```

case(state)
0 :cnt_top<='C5;
1 :cnt_top<='H2;
2 :cnt_top<='H1;
3 :cnt_top<='C5;
4 :cnt_top<='C5;
5 :cnt_top<='H1;
6 :cnt_top<='H2;
7 :cnt_top<='H3;
8 :cnt_top<='H2;
9 :cnt_top<='H1;
10 :cnt_top<='H2;
11 :cnt_top<='C5;
12 :cnt_top<='H2;
13 :cnt_top<='H1;
14 :cnt_top<='C5;
15 :cnt_top<='C5;
16 :cnt_top<='H1;
17 :cnt_top<='H2;
18 :cnt_top<='H3;
19 :cnt_top<='H2;
20 :cnt_top<='H1;
21 :cnt_top<='H2;
22 :cnt_top<='C5;
23 :cnt_top<='H2;
24 :cnt_top<='H1;
25 :cnt_top<='C5;
26 :cnt_top<='C5;
27 :cnt_top<='H1;
28 :cnt_top<='H3;
29 :cnt_top<='C5;
30 :cnt_top<='C6;
31 :cnt_top<='C5;
32 :cnt_top<='C5;
33 :cnt_top<= 0;
34 :cnt_top<='C1;
35 :cnt_top<='C2;
36 :cnt_top<='C2;
37 :cnt_top<='C1;
38 :cnt_top<='C2;
39 :cnt_top<='C3;
40 :cnt_top<= 0;
41 :cnt_top<='C3;
42 :cnt_top<='C5;

```

```

43 :cnt_top<='C6;
44 :cnt_top<='C7;
45 :cnt_top<='C6;
46 :cnt_top<='C5;
47 :cnt_top<='C3;
48 :cnt_top<='C2;
49 :cnt_top<='C2;
50 :cnt_top<='C1;
51 :cnt_top<='C2;
52 :cnt_top<='C2;
53 :cnt_top<='C3;
54 :cnt_top<='C1;
55 :cnt_top<='C1;
56 :cnt_top<= 0;
57 :cnt_top<= 0;
58 :cnt_top<='C1;
59 :cnt_top<='C3;
60 :cnt_top<='C5;
61 :cnt_top<='C6;
62 :cnt_top<='C5;
63 :cnt_top<='C5;
64 :cnt_top<= 0;
65 :cnt_top<='C1;
66 :cnt_top<='C2;
67 :cnt_top<='C2;
68 :cnt_top<='C1;
69 :cnt_top<='C2;
70 :cnt_top<='C3;
71 :cnt_top<= 0;
72 :cnt_top<='C3;
73 :cnt_top<='C5;
default;;
endcase
end
reg [26:0] cnt;
always@(posedge clk or negedge rst)
if(!rst)
cnt<=0;
else if(cnt<50_000_000/cnt_top-1)
cnt<=cnt+1;
else
cnt<=0;
always@(posedge clk or negedge rst)
if(!rst)

```

```

 AUD_PWM<=0;
 else
 AUD_PWM<=(cnt<cnt_top/2)?1:0;

endmodule

```

## AUDIO.c

```

#define PWM_OUT 0x80140000

#define SegEn_ADDR 0x80001038
#define SegDig_ADDR 0x8000103C

#define RPTC_CNTR 0x80001200
#define RPTC_HRC 0x80001204
#define RPTC_LRC 0x80001208
#define RPTC_CTRL 0x8000120c

#define IO_LEDR 0x80100000
#define IO_SW 0x80100008

#define READ_IO(dir) (*(volatile unsigned *)dir)
#define WRITE_IO(dir, value) { (*(volatile unsigned *)dir) = (value); }

unsigned int getshow(unsigned int h, unsigned int m, unsigned int s) {
 unsigned int show = 0;
 unsigned int hshow = 0, mshow = 0, sshow = 0;
 unsigned int high = 0, low = 0;

 high = h / 10;
 low = h % 10;
 hshow |= high;
 hshow <<= 4;
 hshow |= low;

 high = m / 10;
 low = m % 10;
 mshow |= high;
 mshow <<= 4;
 mshow |= low;

 high = s / 10;
 low = s % 10;

```



```

 sshow |= high;
 sshow <<= 4;
 sshow |= low;

 show |= hshow << 24;
 show |= mshow << 12;
 show |= sshow;

 return show;
}

int main (void) {
 static int dayt[][13]={
 {0,31,28,31,30,31,30,31,31,30,31,30,31},
 {0,31,29,31,30,31,30,31,31,30,31,30,31}
 };
 struct date{
 int year;
 int month;
 int day;
 };
 int counter_value;
 unsigned int h = 0, m = 0, s = 0, show = 0 ,show1 = 0, control=0 ,control1 = 0;
 unsigned int seth = 25, setm = 61;
 unsigned int noisy = 0;
 unsigned int hh = 0xFF, mm = 0xff, ss = 0xff;
 unsigned int sw;

 unsigned int alarm_hour = 99; // 闹钟小时，初始设置为无效值
 unsigned int alarm_minute = 99; // 闹钟分钟，初始设置为无效值
 unsigned int alarm_second = 99;
 unsigned int alarm_read = 0;
 unsigned int date_read = 0;
 unsigned int date_display = 0;
 int leap;

 WRITE_IO(PWM_OUT, 0);
 WRITE_IO(SegEn_ADDR, 0x24);
 WRITE_IO(RPTC_LRC, 0x2FFFFFFF);

 struct date date_;
 while (1) {
 sw = READ_IO(IO_SW);
 control = sw & 0xf;

```

```

control1 = sw & 0x8000;
switch(control) {
 case 0:
 leap=((date_.year%4==0)&&(date_.year%100!=0))||((date_.year%400
==0));

 alarm_read = 0;
 date_read = 0;
 date_display = 0;
 if(hh!=0xFF || mm!=0xFF || ss!=0xFF) {
 h = hh;
 m = mm;
 s = ss;
 hh = 0xFF;
 mm = 0xFF;
 ss = 0xFF;
 }
 s++;
 if(s==60) {
 s = 0;
 m++;
 }
 if(m==60) {
 m = 0;
 h++;
 }
 if(h==24) {
 h = 0;
 date_.day += 1;
 if (date_.day == dayt[leap][date_.month] + 1){
 date_.day = 0;
 date_.month += 1;
 if (date_.month == 13){
 date_.month = 0;
 date_.year += 1;
 }
 }
 }
 break;
 case 1:
 h = (sw >> 10) & 0x3f;
 m = (sw >> 4) & 0x3f;
 s = 0;
 break;

```

```

case 2:
 if(hh==0xff && mm==0xff && ss==0xff) {
 hh = h;
 mm = m;
 ss = s;
 }
 alarm_read = 1;
 alarm_hour = (sw >> 10) & 0x3f;
 alarm_minute = (sw >> 4) & 0x3f;
 alarm_second = 0;
 break;
case 4:
 if(hh==0xff && mm==0xff && ss==0xff) {
 hh = h;
 mm = m;
 ss = s;
 }
 h = 0;
 m = (sw >> 10) & 0x3f;
 s = (sw >> 4) & 0x3f;
 break;
case 6:
 if(hh==0xff && mm==0xff && ss==0xff) {
 hh = h;
 mm = m;
 ss = s;
 }
 // 启动闹钟
 show1 = 1;
 break;
case 8:
 if(hh==0xff && mm==0xff && ss==0xff) {
 hh = h;
 mm = m;
 ss = s;
 }
 date_display = 1;
 break;
case 12:
 if(h==0 && m==0 && s==0) {
 noisy = 30;
 h = 99;
 m = 99;
 s = 99;
 }

```

```

 }
 else if(h==99 && m==99 && s==99) {
 ;
 }
 else {
 if(s>0) {
 s--;
 }
 else if(m>0) {
 m--;
 s = 59;
 }
 }
 break;
}
switch(control1) {
 case 0x8000:
 if(hh==0xff && mm==0xff && ss==0xff) {
 hh = h;
 mm = m;
 ss = s;
 }
 date_.year = (sw >> 9) & 0x3f;
 date_.month = (sw >> 5) & 0xf;
 date_.day = sw & 0x1f;
 date_read = 1;
 break;
}
if(h==seth && m==setm) {
 noisy = 10;
 seth = 25;
 setm = 61;
}
if(hh!=0xFF || mm!=0xFF || ss!=0xFF) {
 ss++;
 if(ss==60) {
 ss = 0;
 mm++;
 }
 if(mm==60) {
 mm = 0;
 hh++;
 }
 if(hh==24) {

```

```

 hh = 0;
 }
}
if (alarm_read == 1){
 show = getshow(alarm_hour, alarm_minute, alarm_second);
}
else if (date_read == 1){
 show = getshow(date_.year, date_.month, date_.day);
}
else if (date_display == 1){
 show = getshow(date_.year, date_.month, date_.day);
}
else {
 show = getshow(h, m, s);
}
WRITE_IO(SegDig_ADDR, show);

if (show1 && h == alarm_hour && m == alarm_minute) {
 noisy = 10;
 show1 = 0; // 重置闹钟设置，避免重复激活
}

if(noisy>0) {
 WRITE_IO(PWM_OUT, 1);
 noisy--;
}
else {
 WRITE_IO(PWM_OUT, 0);
}
WRITE_IO(RPTC_CTRL, 0xC0);
WRITE_IO(RPTC_CTRL, 0x21);
while(1){
 counter_value = READ_IO(RPTC_CTRL);
 counter_value = counter_value & 0x40;
 if(counter_value!=0){
 break;
 }
}
}
return(0);
}

```

## 指导教师评定意见

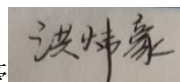
---

### 一、原创性声明

本人郑重声明本报告内容，是由本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明的内容外，本报告不包含任何其他个人或集体已经公开的发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：洪炜豪



### 二、对课程实验的学术评语（教师填写）

### 三、对课程实验的评分（教师填写）

评分项目 (分值)	报告撰写 (40 分)	实验过程 (60 分)	最终评定 (100 分)
得分			

指导教师签字：\_\_\_\_\_