黑金开发板之音频模块实验

Rev. 1.00

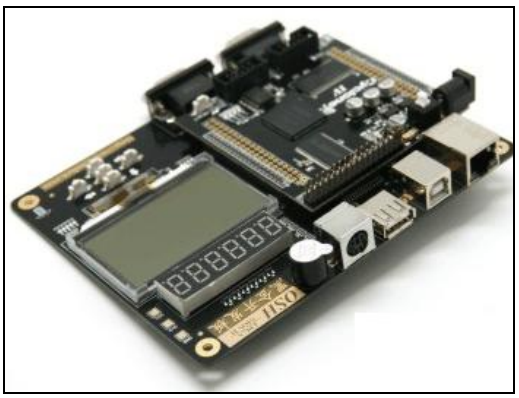
版本记录

版本	时间	作者	描述
Rev1.00	2015-3-10		First Release

一、实验前准备

此手册为用户介绍如何在黑金开发板上实现语音和播放的实验及 SD 卡音乐播放的试验。在实验之前用户需要准备以下的开发板和配件:

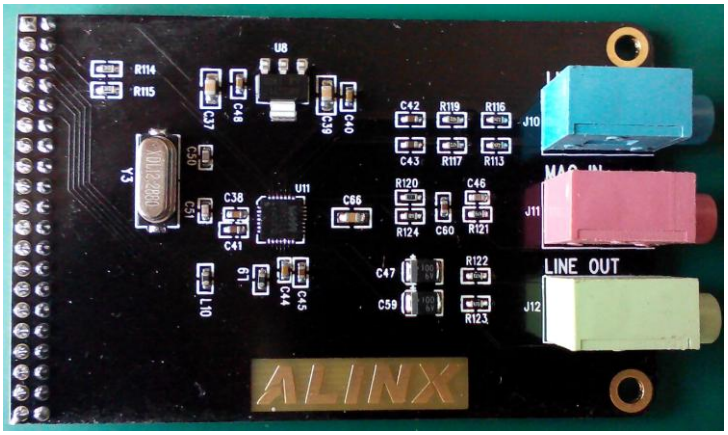
1. 黑金开发板套件: AX301 学生版套件或 AX415 旗舰版开发板套件



AX415 旗舰版

AX301 学生版

2. 音频模块：AN831 音频模块

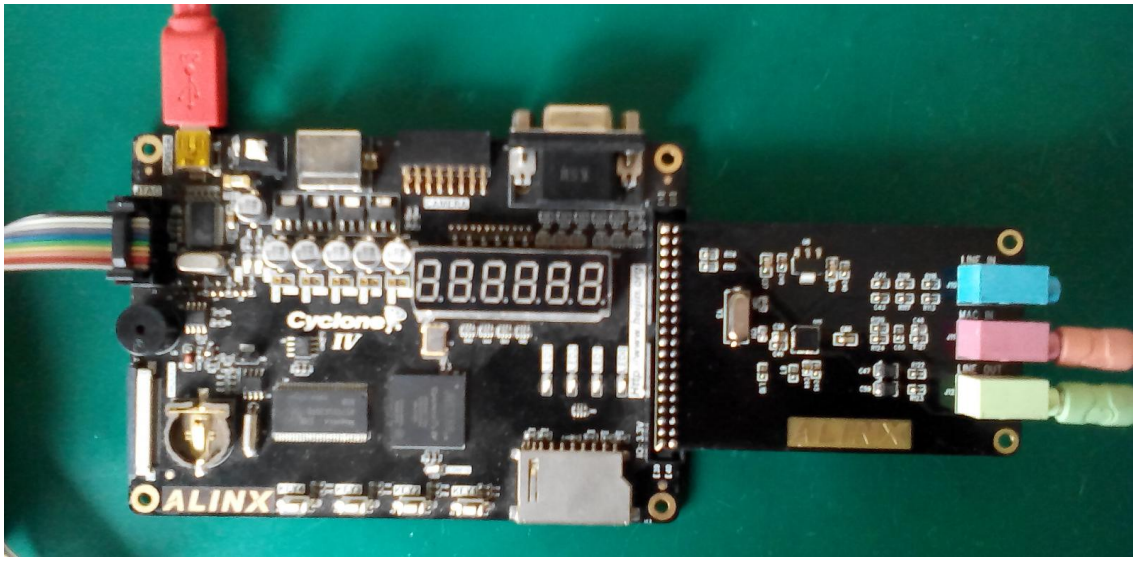


3. SD 卡：SD HC 卡 (对于老的 1.0 标准的 SD 卡，我们这里的程序不做支持

4. 带麦克风的耳机

二、录音及播放例程实验

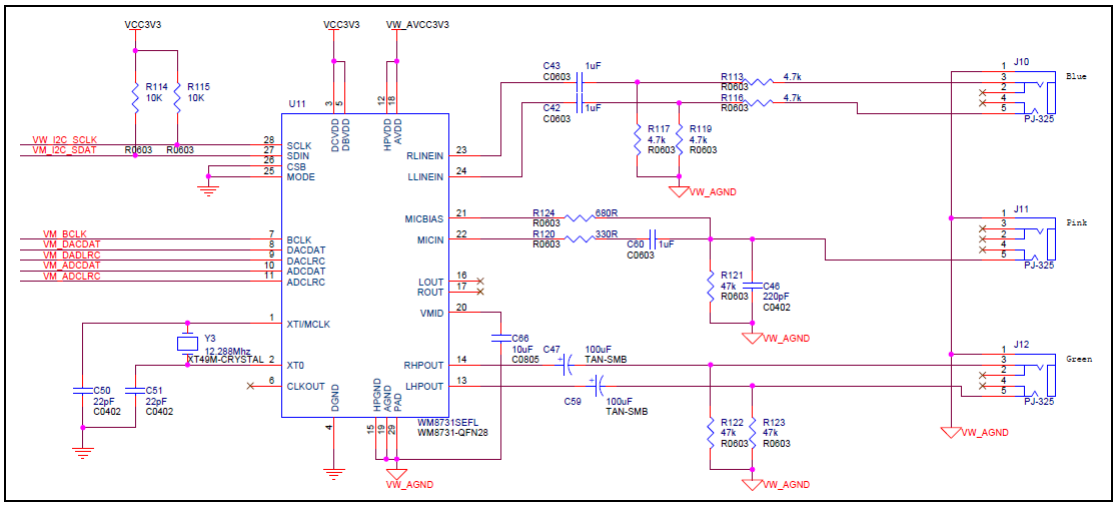
音频模块是跟黑及开发板上的 40 针扩展口相连实现音频数据通信。音频模块和 AX301 学生版开发板连接后如下图所示:



音频模块上有三个音频连接器,其中粉色的接口为麦克风输入;绿色的接口为耳机输出;蓝色的接口为音频输入,用于连接 DVD 等音频输出口。本实验将实现音频模块和 FPGA 之间的数据通信, 通过音频模块把麦克风输入的语音数据存储到开发板上的 SDRAM 存储器里, 再把音频数据发送给音频模块,从耳机接口进行语音的播放,从而实现录音和播放的功能。

1 音频硬件介绍

AN871 音频模块使用 WOLFSON 公司的 WM8731 芯片实现声音信号的 A/D 和 D/A 转换功能。以下为 AN831 音频模块的硬件电路：



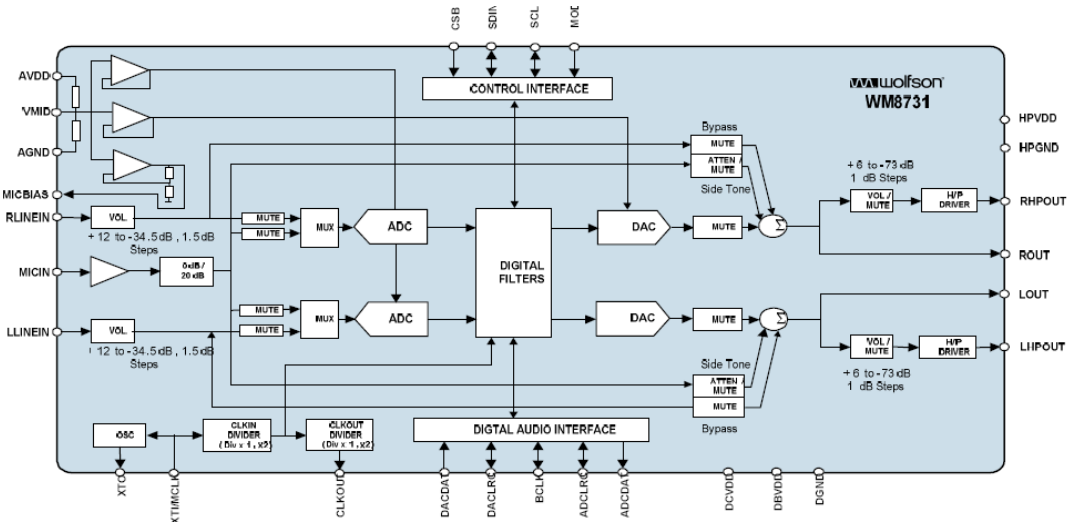
音频模块上的40PIN连接器（J3）的管脚定义如下：

J3 管脚名	信号名	J3 管脚名	信号名
1	地	2	5V电源
3	VM_I2C_SCLK	4	VM_I2C_SDAT
5	VM_DACDAT	6	VM_BCLK
7	VM_ADCDAT	8	VM_DADLRC
9	VM_ADCLRC	10	空脚
11	空脚	12	空脚
13	空脚	14	空脚
15	空脚	16	空脚
17	空脚	18	空脚
19	空脚	20	空脚
21	空脚	22	空脚
23	空脚	24	空脚
25	空脚	26	空脚
27	空脚	28	空脚
29	空脚	30	空脚
31	空脚	32	空脚
33	空脚	34	空脚
35	空脚	36	空脚
37	地	38	地
39	3.3V电源	40	3.3V电源

2 WM8731 配置和时序

我们来简单的介绍一下音频模块 AN831 用到的音频编/解码芯片 WM8731。该芯片在本设计中主要完成声音信号在采集和回放过程中的 A/D 和 D/A 转换功能。该芯片的 ADC 和 DAC 的采样频率为 8KHz 到 96KHz 可调，可转换的数据

长度为 16-32 位可调。WM8731 的内部有 11 个寄存器。该芯片的初始化以及工作时的工作状态和功能都是通过以I2C总线方式对其内部的这11个寄存器进行相应的配置来实现的。本设计中 WM8731 工作于主模式，采样频率设为 48KHZ,转换的数据位长度为 16 位。WM8731 的音频接口可通过编程设置为 I2S 或 DSP/PCM 接口形式。

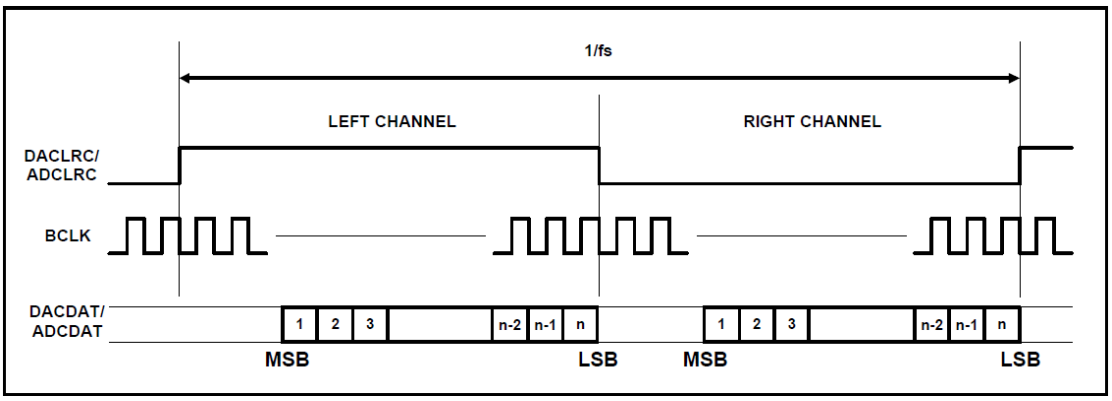


在本实验中 FPAG 和 WM8731 的控制和数据通信将用到 I2C 和 I2S 总线接口。FPGA 通过 I2C 接口配置 WM8731 的寄存器，通过 I2S 总线接口来进行音频数据的通信。关于 I2C 接口，我们已经在前面部分讲过，下面我们主要来了解 I2S 的音频通信接口。

音频接口 I2S：

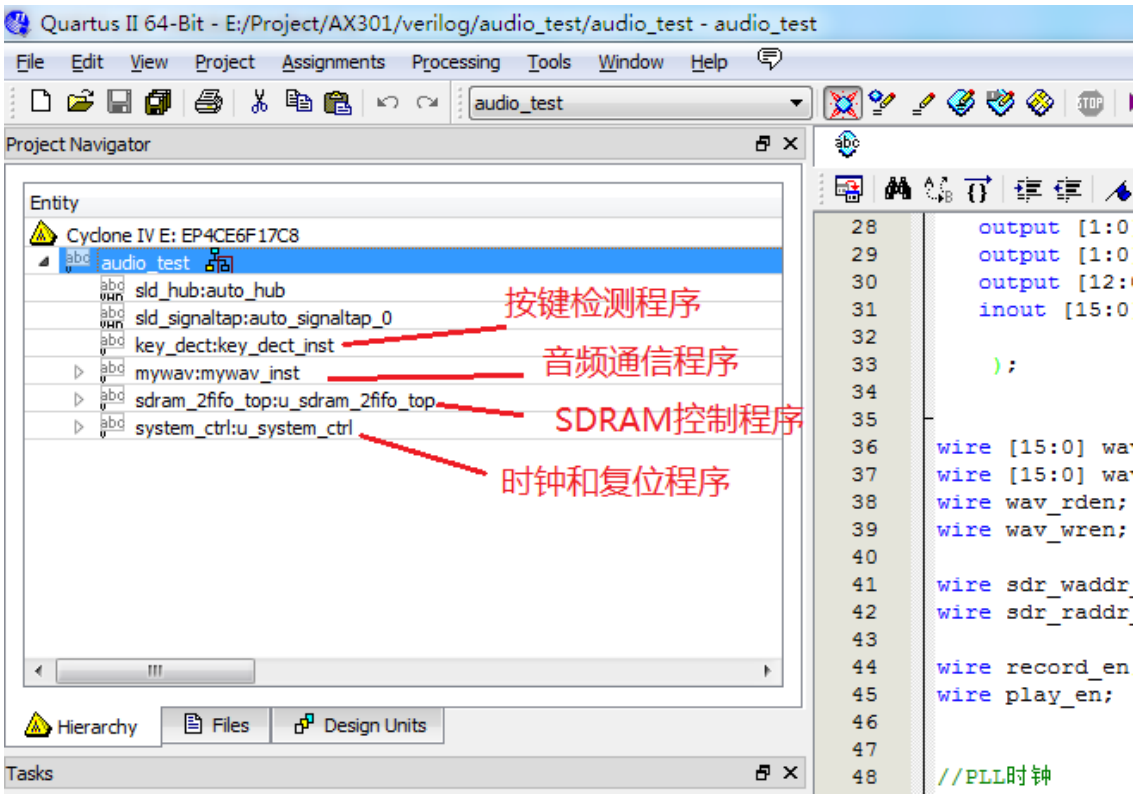
WM8731 的数字音频接口有 5 根引脚，分别为：BCLK(数字音频位时钟)、DACDAT(DAC 数字音频数据输入)、DACLRC(DAC 采样左/右声道信号)、ADCDAT(ADC 数字音频信号输出)、ADCLRC(ADC 采样左/右声道信号)。

在本设计中 FPGA 为从设备，WM8731 为主设备。ADCDAT、DACDAT、ADCLRC 和 DACLRC 与位时钟 BCLK 同步，在每个 BCLK 的下降沿进行一次数据传输。BCLK、DACDAT、DACLRC、ADCLRC 为 WM8731 的输入信号。ADCDAT 为 WM8731 的输出信号。FPGA 和 WM8731 的 I2S 的通信右对齐的时序图如下图所示:



3 程序设计

本实验的功能是程序检测按键 KEY1 是否按下，如果 KEY1 按下，录音开始；如果 KEY1 松开，录音结束，播音开始。就像我们手机上使用的微信那样，按住开始说话，放开录音完成。本程序设计包含四大部分：SDRAM 的读写控制，音频控制和通信，按键检测和时钟复位延迟模块。以下是为 AX301 开发板完成的工程的结构图：



1). Sdram 控制程序设计

SDRAM 控制程序包含两个子程序:一个是 SDRAM 读写控制程序(sdram_top .v)， 另一个为 FIFO 控制程序(dcfifo_ctrl.v)。

SDRAM 读写控制程序说明：

SDRAM 读写控制程序(sdram_top .v)和 3 个子模块(sdram_ctrl.v, sdram_cmd.v, sdram_wr_data.v) s 实现了 sdram 的初始化 ,用户接口的读写命令解析，sdram 的突发读写，自刷新和预充电等操作控制。

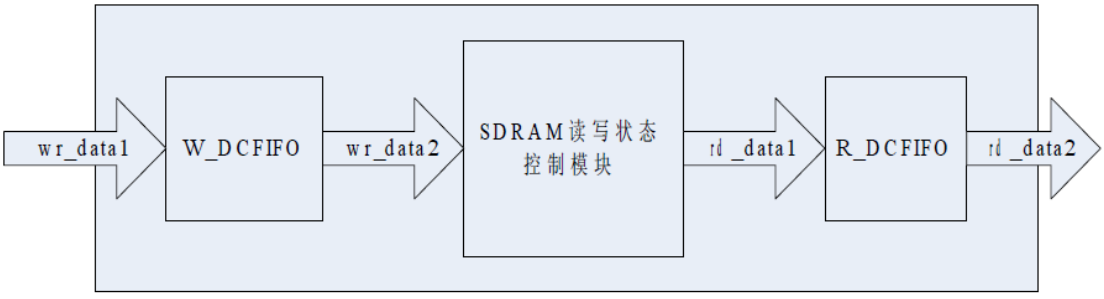
其中 sdram_ctrl 模块实现 SDRAM 的初始化，60ms 的自刷新，用户读写请求命令解析，使用状态机和计数器生成不同 SDRAM 操作的状态位。

sdram_cmd 模块根据 sdram_ctrl 模块中产生的状态机 init_state 和 work_state 来产生各种 SDRAM 的控制或 burst 读写命令。

sdram_wr_data 模块是 SDRAM 读写双向数据控制模块，在写 SDRAM 时，把数据传输到 SDRAM 的数据总线上，在读 SDRAM 时，把 SDRAM 总线上的数据传输给用户接口。

FIFO 控制程序说明：

Dcfifo_ctrl.v 模块用于对读 FIFO 和写 FIFO 的控制和 SDRAM 的读写命令和读写地址的产生。本实验中，向 SDRAM 写入的数据首先存放在写 FIFO 中，从 SDRAM 中读出的数据首先存放在读 FIFO 中。



当写 FIFO 的数据内数据大于一个 SDRAM burst 长度(256)的时候产生 Sdram 写命令。

```
190     else if(sdram_init_done == 1'b1)
191       begin                 //写入优先，带宽内防止数据丢失
192       if(wrf_use >= wr_length && frame_write_done == 1'b0)  //
193         begin                       //wrfifo满突发长度
194         sdram_wr_req <= 1;          //写sdarm使能
195         sdram_rd_req <= 0;          //读sdram空闲
196         end
197       else if(rdf_use < rd_length && data_valid_r == 1'b1 && frame_write_done == 1'b1)
198         begin               //rdfifo满突发长度
199         sdram_wr_req <= 0;          //写sdram空闲
```

当读 FIFO 的数据长度小于一个 SDRAM burst 长度(256)的时候产生 Sdram 读命令。

```
196         end
197       else if(rdf_use < rd_length && data_valid_r == 1'b1 && frame_write_done == 1'b1)
198         begin              //rdfifo满突发长度
199         sdram_wr_req <= 0;       //写sdram空闲
200         sdram_rd_req <= 1;       //读sdram使能
201         end
202       else
```

2). 音频通信控制程序

音频部分包含一个主程序(mywav.v)和四个子程序 ,四个子程序分别是音频接收程序(sinwave_store.v), 音频播放程序(sinwave_gen.v), WM8731 寄存器配置程序(reg_config.v) 和复位延迟程序（ reset_delay.v ）。另外寄存器配置程序(reg_config.v)还调用了 iic 的通信程序 i2c_com.v。

音频接收程序 sinwave_store.v 说明

程序通过判断 bclk 输入时钟的上升沿来采样音频 adcdat 输入的数据，串行转化为 16bit 的并行数据并产生 SDRAM 写请求信号。

音频播放程序 sinwave_gen.v 说明

程序通过判断 bclk 输入时钟的下降沿来移位输出 64bit 的音频数据到 dacdat 引脚 ,因为 1 个 fs 的音频数据为 64bit, 程序需要产生 4 个读 SDRAM 的信号。

WM8731 寄存器初始化程序 reg_config.v 说明

此程序上电后会通过 I2C 总线对 WM8731 芯片的寄存器的初始化配置，关于 WM8731 详细的寄存器介绍，请大家参考芯片的 datasheet。

IIC 的通信程序 i2c_com.v 说明

IIC 通信程序把外部过来的数据按时序移位输出到外部的 IIC 总线上，从而实现 IIC 的数据写的功能。

复位延迟模块 reset_delay.v 说明

这是软件上电后复位的一种方法，目的是为了上电后等待一段时间后再进行 WM8731 的寄存器配置。

3）. 按键检测程序

程序会检测按键 KEY1 是否按下或按键是否松开。如果检测到 KEY1 按键按下，录音使能信号为高，SDRAM 写地址清 0 ；如果检测到 KEY1 按键松开，播放使能信号为高，SDRAM 读地址清 0 。

4). 系统控制模块

system_ctrl.v 程序调用 PLL 产生 100Mhz 的 SDRAM 的时钟。另外调用 system_delay 模块产生系统级的复位信号。

4 下载和测试

连接耳机和音频模块。耳机的接口注意不要插错，耳机的粉色插头插到音频模块的粉色接口上，耳机的绿色插头插到音频模块的绿色接口上。

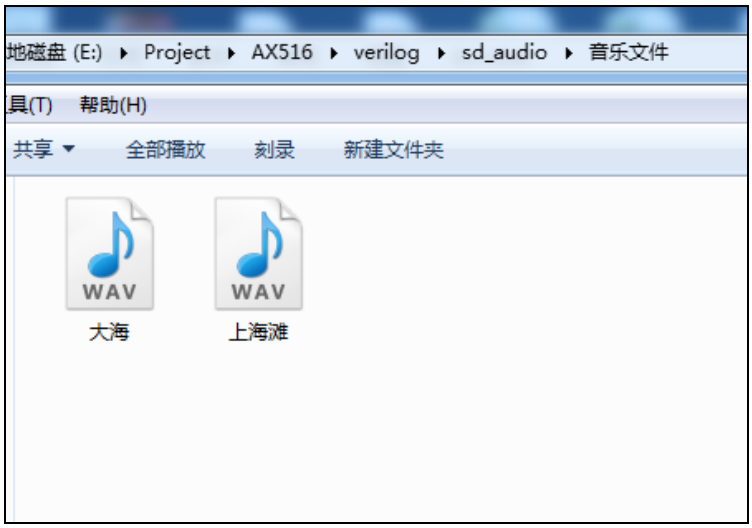
编译工程生成 audio_test.sof 文件并下载 bit 文件到 FPGA 里。这时我们按住开发板上的 KEY1 再对着麦克风说上一段话 释放 KEY1 后就能在耳机里听到刚才您自己说的话了。

三、SD 卡音乐播放例程实验

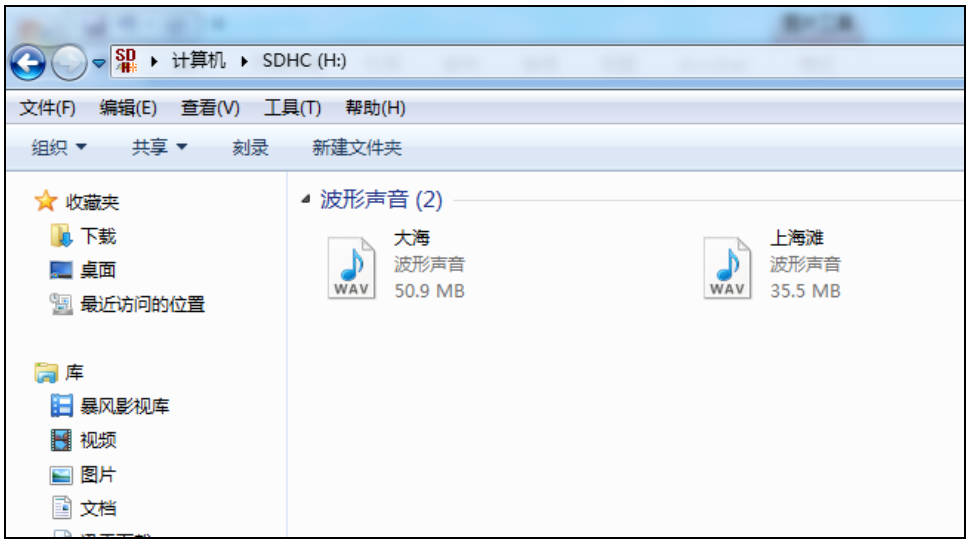
1 音乐文件

在本实验之前我们需要在 SD 卡里存入几首 wav 格式的音乐文件注意这里的.wav 音乐文件格式需要为 16 位，采样频率为 48kHz 的，这跟 WM8731 的寄存器设置有关。文件格式可以右击.wav 文件，选择属性来查看。

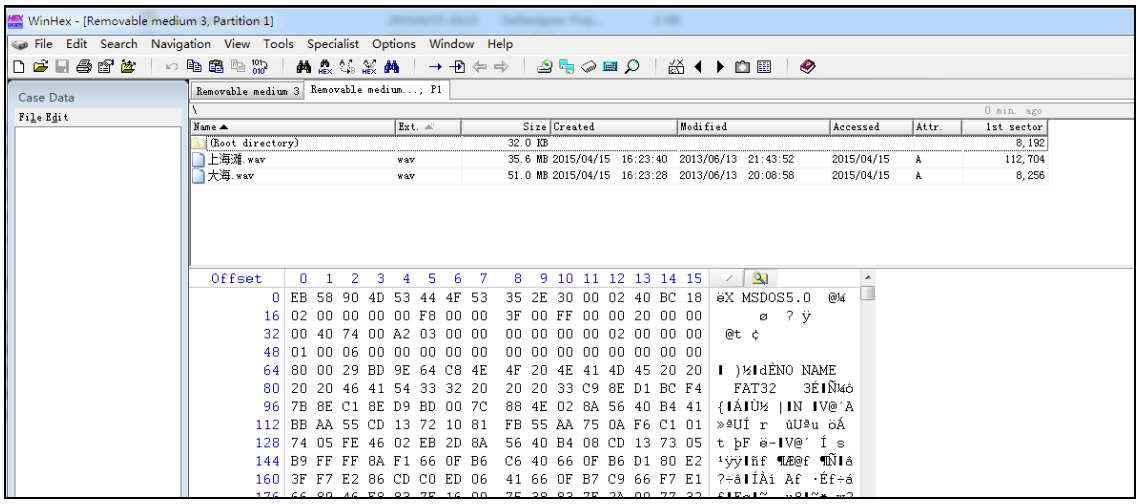
关于音乐文件,用户可以从网上下载 wav 格式的音乐，再通过软件转化成 16 位,采样频率为 48Khz 格式的 wav 文件,或者直接使用我们提供的 wav 音乐文件做实验。在我们提供的光盘实验例程的目录下已经准备了两首歌曲供大家享用。



我们先用电脑格式化一下 SD 卡，再把这两首歌拷贝到 SD 卡的根目录，向 SD 卡的根目录存放两首音乐文件后如下：



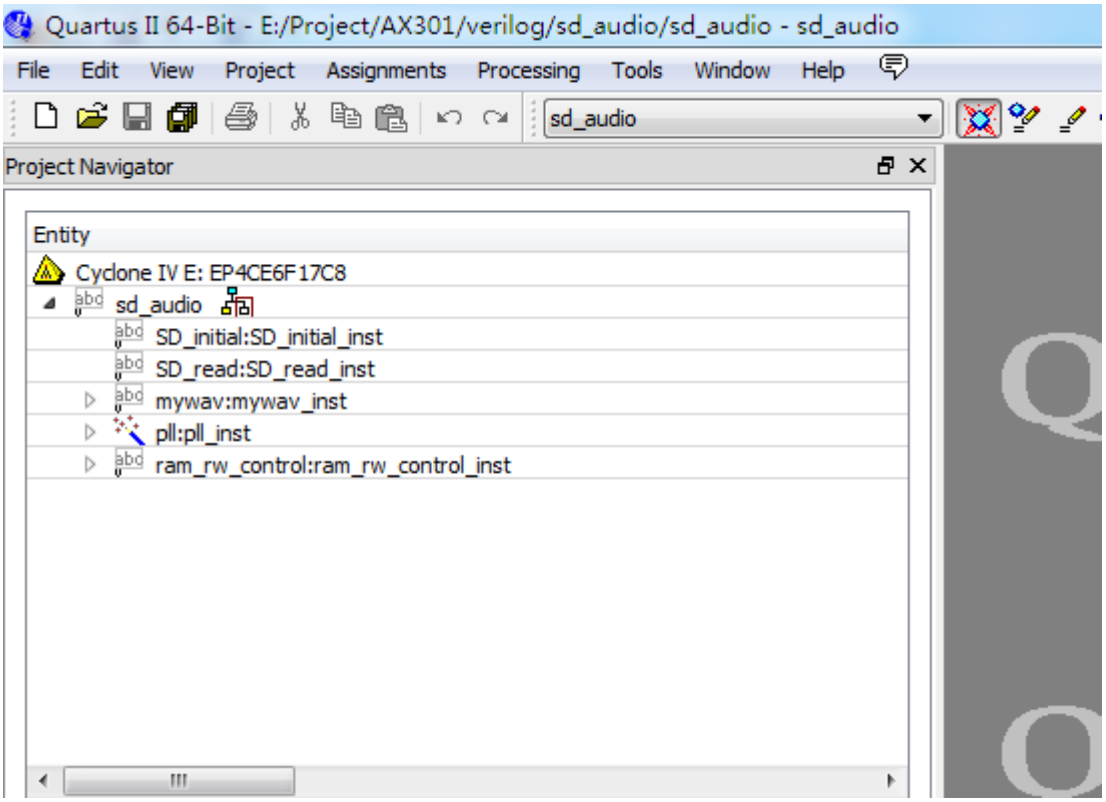
我们再用 winhex 工具来查看一下这两首歌曲在 SD 卡的 Sec 地址，这个地址就是我们在下面编写读 SD 卡程序的时候的起始 Sec 地址。



在 Winhex 窗口我们可以看到,SD 卡的根目录的 Sec 地址为 8192。歌曲大海的 Sec 地址为 8256,我们在程序里就从 8192+8256=16488 的地址开始读数据就可以了。

2 程序设计

整个项目 sd_audio 是由一个顶层模块 sd_audio 程序和几个子模块组成(SD 卡的初始化程序 sd_initial.v，SD 卡的读程序 sd_read.v，WM8731 的音频程序 mywav.v，ram 数据存储控制程序 raw_rw_control.v)。以下是为 AX301 开发板完成的工程的结构图：

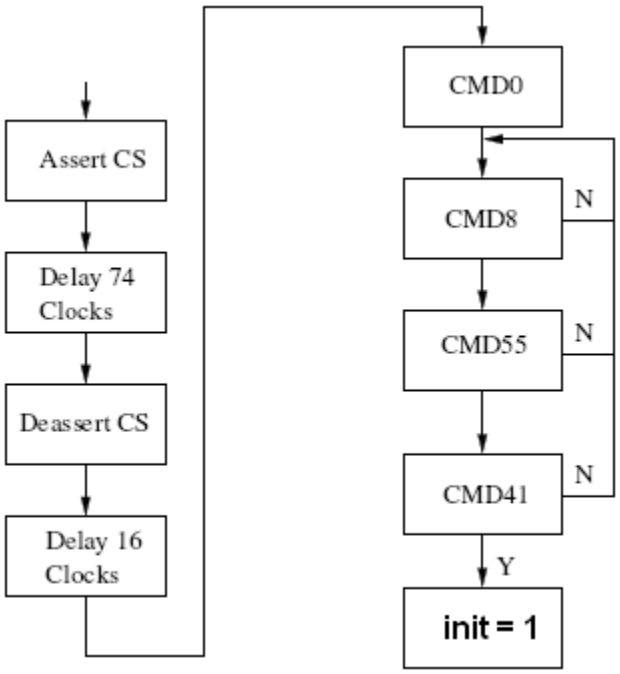


关于 WM8731 的 I2C 的寄存器配置和 I2S 语音播放我们在”录音和播放例程”中给大家讲过，用户直接拷贝这些程序在本实验使用就可以了。

在本实验，SD 卡读出的数据是存放在程序定义的 ram 数组里，ram 的地址和数据读写控制在 raw_rw_control.v 程序里完成。下面我们将对 SD 卡和 ram 控制的程序设计作简单介绍和讲解。

1）SD 的初始化程序设计

SD 卡上电后需要对其初始化操作，S 初始化过程和 SD 卡的相关命令请参考 SD 卡 2.0 的协议标准，初始化流程如下：



SD卡初始化流程

具体流程说明如下：

- 片选 CS 低电平选中 SD 卡。
- 片选 CS 高电平释放 SD 卡。
- 发送 CMD0 命令给 SD 卡，并确认返回是否为 0x01。
- 发送 CMD8 命令给 SD 卡，CMD8 是 SD2.0 才有的命令。并确认返回[bit19:bit16]是否为 0001(2.7V-3.6V)。
- 发送 CMD55 命令给 SD 卡，确认是否返回 0x01。

- 发送 ACMD41 命令给 SD 卡，确认是否返回 0x00。
- 如果初始化成功，init 变量置 1。初始化未成功,重新发送 CMD8, CMD55 和 ACMD41 命令。

2）SD 的读程序设计

当 SD 卡初始化成功后,开始读取图像的数据到 SDRAM 中。程序发送 CMD17 单块读命令给 SD 卡，连续读取 512 个字节，直到读完一幅图像。

具体流程说明如下：

- 发送 CMD17 单块读命令给 SD 卡并等待应答。
- 等待从 SD 卡接收数据,如果检测到数据的起始位(0),则开始接收数据。
- 接收 512 个字节数据,把从 SPI 接收到的串行数据转化成 8bit 数据的字节,每个字节输出一个数据字节有效信号。
- 接收完 512 个字节的数据后判断是否读到图像的最后的数据。

3）ram 控制的程序设计

程序中定义了一个 8192 长度的一个 myram 寄存器用于存放 SD 卡读出的音乐数据，程序中读写实现乒乓结构，读前 4096 空间的数据时，程序可以写后面的 4096 空间的数据。同样在写前 4096 空间的数据时，程序可以读后面的 4096 空间的数据程序。因为 SD 卡的读的速度远远大于音乐播放的速度，所以使用乒乓结构能够满足音乐数据 SD 卡读取并同时播放的功能。

上电后会首先读取 4096 个数据预先存放在 ram 寄存器中。程序接收到 WM8731 的音频程序 mywav.v 发来的数据读请求后 ,把 ram 寄存器里的 2 个字节的数据发送给 mywave.v 程序，ram 的读地址增加。另外程序判断读地址，当读 ram 的地址为 2 或 4096 的时候，产生 SD 卡读，读取 4096 个数据。

其中代码中下面的两行音乐文件在 SD 卡的起始 Sector 地址和 Sector 的长度。

```
27    parameter SADDR=32'd16488;
28    parameter OADDR=32'd15269887;
29
```

3 下载和测试

把装有音乐文件的 SD 卡插入开发板并插上音频模块，再把耳机接到音频模块的绿色音频输出接口上，在 Quartus 软件中下载 sof 文件后，我们就可以在耳机里听到美妙的音乐了。