



ALINX AX415 Audio Module User Manual

[Home](#) » [ALINX](#) » ALINX AX415 Audio Module User Manual 



AX415 Audio Module
User Manual



Audio Module AN831

Contents

- 1 Part 1: Preparation before Experiment
- 2 Part 2: Recording and playback routine experiment
- 3 Part 3: SD card music player routine
- 4 Documents / Resources
 - 4.1 References
- 5 Related Posts

Part 1: Preparation before Experiment

This instruction manual introduce how to implement voice and playback experiment, and SD card music play on the ALINX serial development Kit, Before the experiment, the user needs to prepare the following development boards and accessories.

1. ALINX serial development Kit: Figure1-1: AX415 FPGA development board or Figure1-2:AX301 FPGA development board.



Figure 1-1: AX415 FPGA board



Figure 1-2: AX301 FPGA board

2. Audio Module Figure1-3: AN831 Audio Module

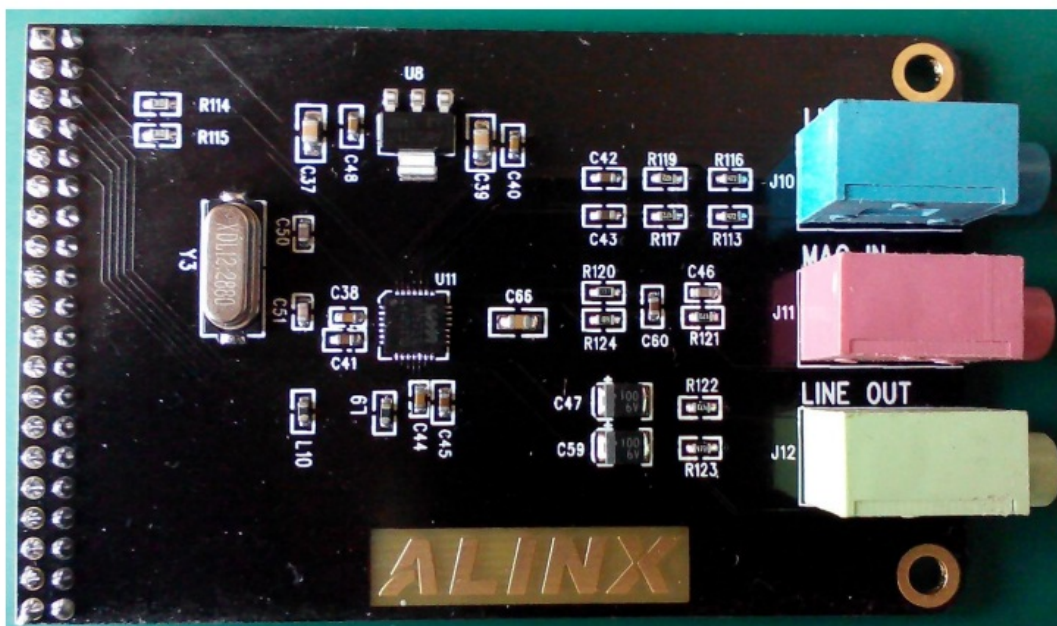


Figure 1-3: AN831 Audio Module

3. SD Card SD HC card The program provided not support the F SD card with Old Version 1.0 standard



Figure 1-4: SD HC card

4. The Headphones with microphone

Part 2: Recording and playback routine experiment

The audio module AN831 is connected to the 40-pin expansion IOs on the ALINX Series FPGA development Kits to implement audio data communication.

Figure 2-1 detailed the connection between the audio module and AX301 FPGA development kit.

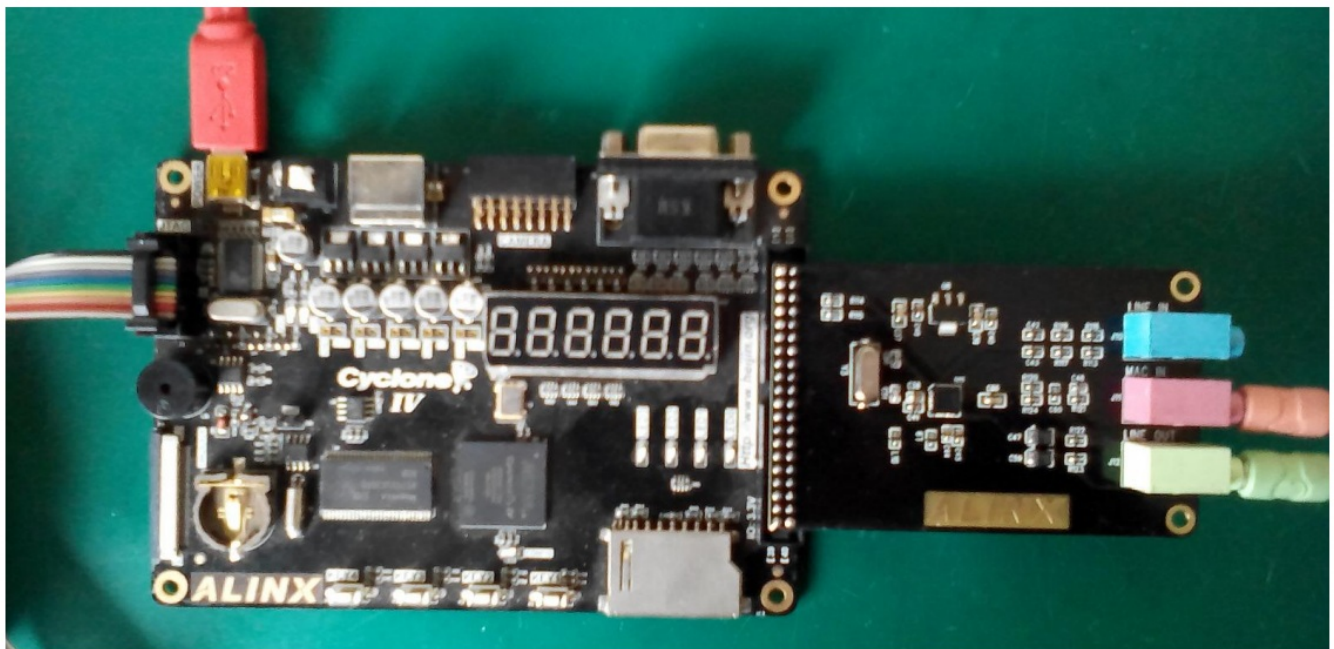


Figure 2-1: The Audio Module AN831 Connected to AX301 FPGA Board

There are three audio connectors on the audio module AN831, the pink interface is the microphone input; the green interface is the headphone output; the blue interface is the audio input, which is used to connect the audio output port such as DVD. This experiment will realize the data communication between the audio module and the FPGA. The voice data input by the microphone is stored in the SDRAM memory on the development board through the audio module, then send the audio data to the audio module, and play the voice from the earphone

interface, thereby realizing the function of recording and playing.

Part 2.1: Audio Module AN831 hardware introduction

Then AN831 Audio Module, use the chip WM8731 of WOLFSON, realize the A/D and D/A conversion of Sound signal. Figure 2-2 detailed the hardware design of the Audio Module AN831

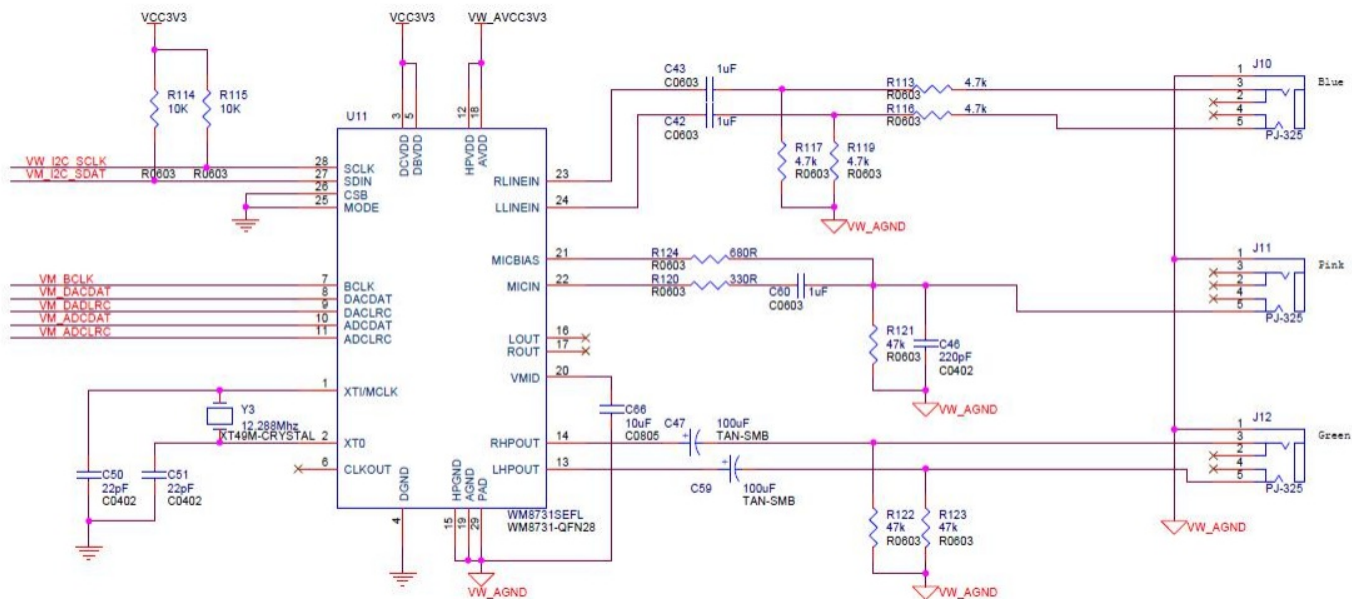


Figure 2-2: The Schematic of Audio Module AN831

The table 2-1 detailed the AN9134 module 40-pin configuration

J3 PIN	PIN Name	J3 PIN
1	Ground	2
3	VM_I2C_SCLK	4
5	VM_DACDAT	6
7	VM_ADCDAT	8
9	VM_ADCLRC	10
11	NC	12
13	NC	14
15	NC	16
17	NC	18
19	NC	20
21	NC	22
23	NC	24
25	NC	26
27	NC	28
29	NC	30
31	NC	32
33	NC	34
35	NC	36
37	Ground	38
39	3.3V Power	40

Table 2-1: AN831 Module 40-pin Configuration

Part 2.2: WM8731 Configuration and Timing

Here briefly introduce the audio encoding/decoding chip WM8731 used in the audio module AN831, which mainly perform A/D and D/A conversion function of the sound signal during recording and playing. The WM8731, stereo 24-bit multi-bit sigma delta ADCs and DACs are used with oversampling digital interpolation and decimation filters. Digital audio input word lengths from 16-32 bits and sampling rates from 8 kHz to 96 kHz are supported. There are 11 registers with 16 bits per register (7 bit address+9 bits of data). The initialization of the chip, the working state, and function during operation are realized by configuring the 11 internal registers in the I2C bus mode. In this designation of audio module AN831, the WM8731 works in the main mode, the sampling frequency is set to 48KHZ, and the converted data bit length is 16 bits. The audio interface of the WM8731 can be programmed to I2S mode or DSP/PCM in mode.

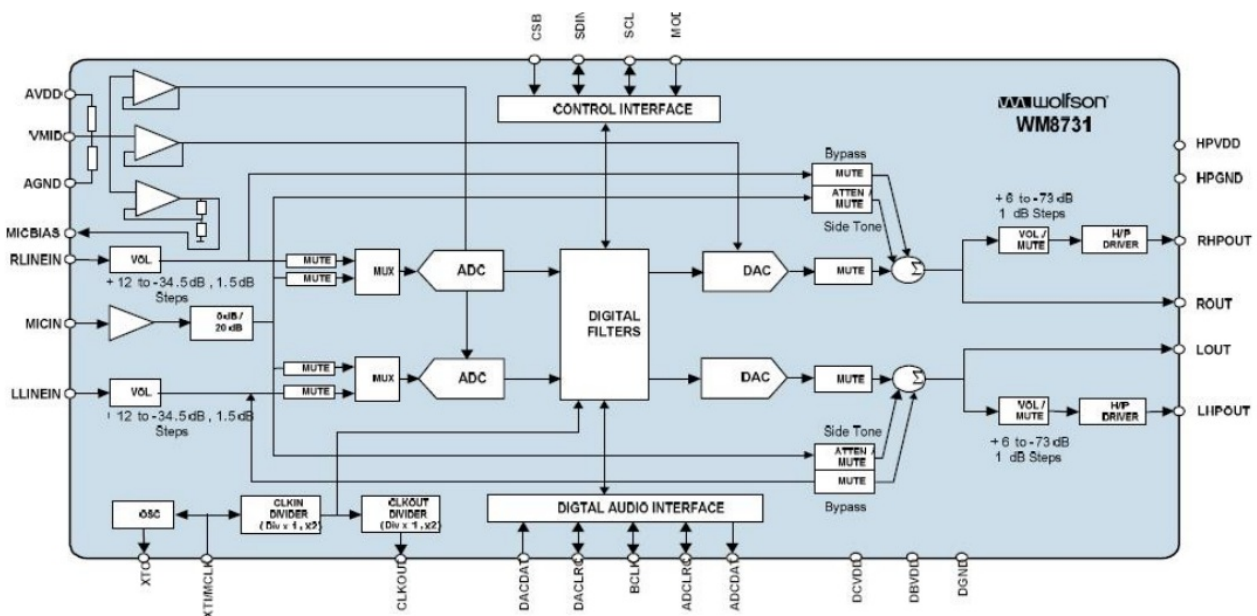


Figure 2-3: Block Diagram of WM8731 chip

In this experiment, the control and data communication of FPG and WM8731 will use I2C and I2S bus interface. The FPGA configures the registers of the WM8731 through the I2C interface, and communicates audio data through the I2S bus interface. About the I2C interface, we briefly introduced before. Here we mainly introduced the audio communication interface I2S.

Audio interface I2S

There are 5 digital audio interfaces on chip WM8731 BCLK(Digital Audio Bit Clock), DACDAT(DAC Digital Audio Data Input), DACLRC(DAC Sample Rate Left/Right Clock), ADCDAT(ADC Digital Audio Data Output), ADCLRC(ADC Sample Rate Left/Right Clock).

In this design, the FPGA is a slave device and the WM8731 is a master device. ADCDAT, DACDAT, ADCLRC, and DACLRC are synchronized with bit clock BCLK. Data transfer on the falling edge of each BCLK. BCLK, DACDAT, DACLRC, ADCLRC are the input signals of WM8731. ADCDAT is the output signal of WM8731. Figure 2-4 detailed the right justified mode of I2S Communication between FPGA and WM8731 chip on audio module AN831.

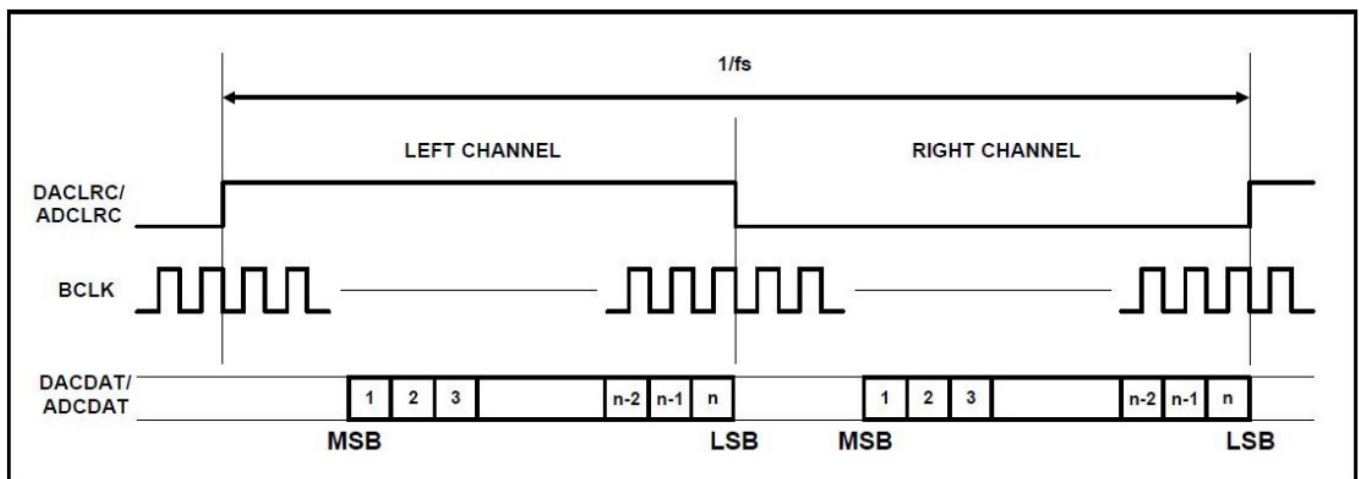


Figure 2-4: Right Justified Mode of I2S Communication

Part 2.3: Programming

The experiment detects whether the button KEY1 is pressed. If KEY1 is pressed, the recording starts; if KEY1 is released, the recording ends and the broadcast starts. Just like the WeChat used on our mobile phone, press and hold to start talking and release to end the recording. This program includes four parts: SDRAM read and write control, audio control and communication, button detection and clock reset delay module. Figure 2-5 is the project navigator of AX301 FPGA Development board.

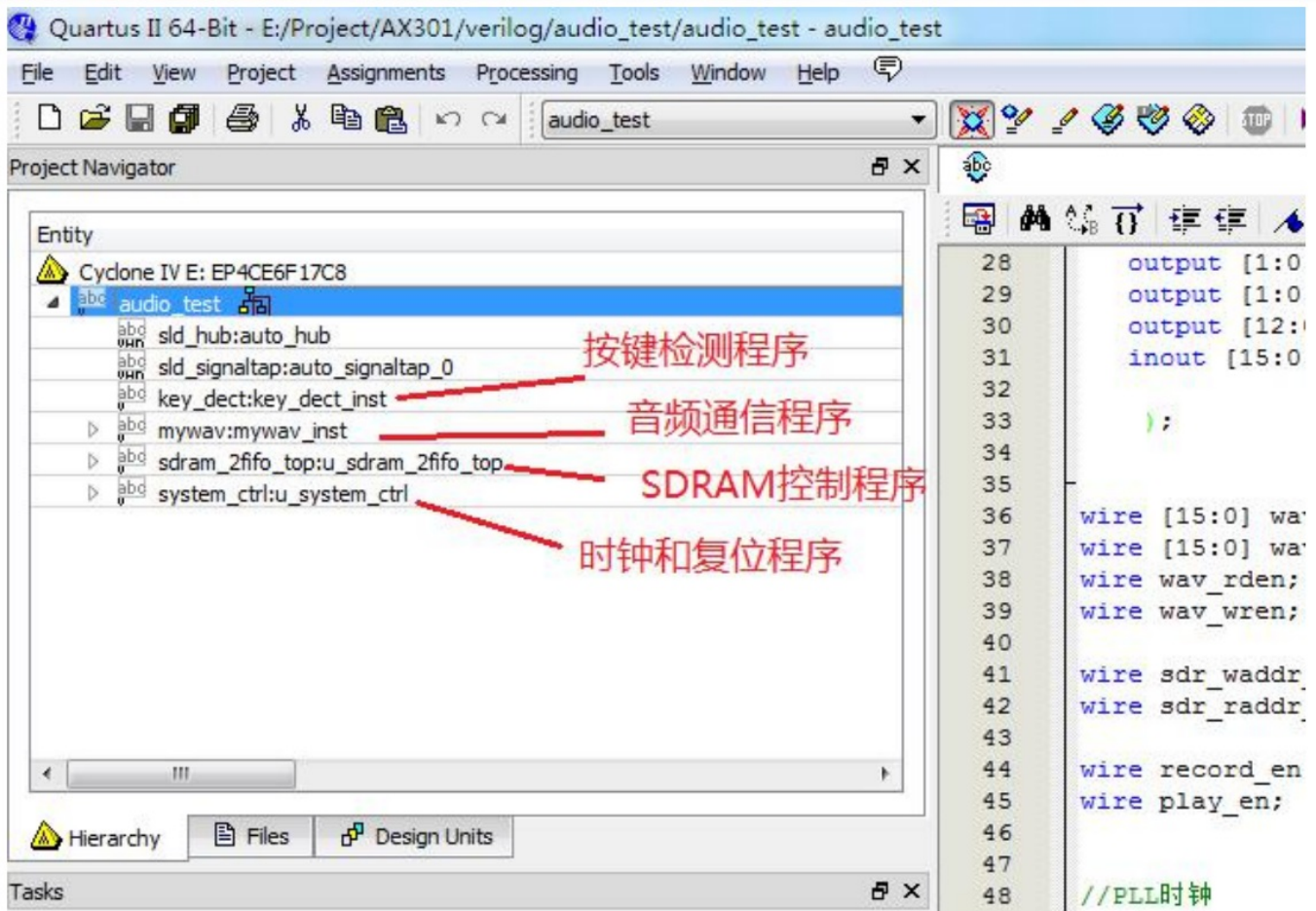


Figure 2-5: The Project Navigator

1. Scram Read and Write Control Program

SDRAM read and write control program include two subroutines : one is SDRAM read and write control document sdram_top.v and another is FIFO control document FIFO

SDRAM read and write control document description:

SDRAM read and write control document (sdram_top.v) and 3 sub modules (dram, dram, dram) initialized the scram, parsed the read and write command of the user interface, burst read and write of dram and pre-charge operation control.

The dram module implements SDRAM initialization, 60ms self-refresh, user read and write request command parsing, and uses status machines and counters to generate status bits for different SDRAM operations.

The dram module generates various SDRAM control or burst read and write commands based on the state machine init_state and work_state generated in the dram module. The dram module is an SDRAM read/write bidirectional data control module. When writing SDRAM, the data is transferred to the SDRAM data bus. When the SDRAM is read, the data on the SDRAM bus is transmitted to the user interface.

FIFO control document description

The FIFO module is used to control read FIFO, write FIFO, and generate the read and write commands and read and write addresses of the SDRAM. In this experiment, the data written to the SDRAM is first stored in the write FIFO, and the data read from the SDRAM is first stored in the read FIFO.

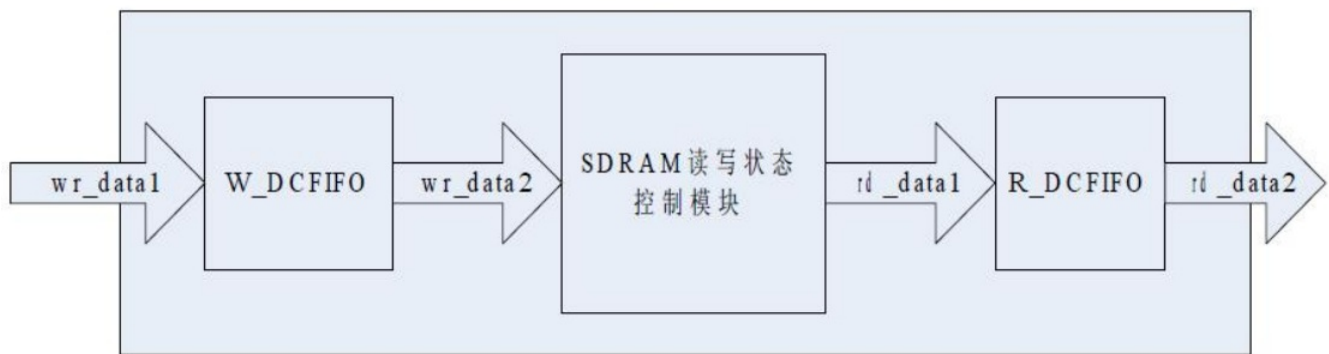


Figure 2-6: SDRAM read and write status control module

Generate SDRAM write command when the data in the write FIFO is greater than the SDRAM burst length (256).

```

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

```

Generate SDRAM read command when the data in the read FIFO is less than the SDRAM burst length (256).

```

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

```

2. Audio Communication Control Program

The audio communication control program consists of a main program (mywav.v) and four subroutines. The four subprograms are the audio receiver (sin wave), the audio player (sin wave), the WM8731 register configuration program (reg_config.v) and the reset delay program (reset_delay.v). In addition, the register configuration program (reg_config.v) also calls the communication program i2c_com.v of IIC.

Audio Receiver Program sin wave Description

The program samples the data input by the audio addax by judging the rising edge of the balk input clock. Serial data is converted to 16-bit parallel data and an SDRAM write request signal is generated.

Audio Player Program sin wave Description

The program shifts output of 64-bit audio data to the ducat pin by judging the falling edge of the balk input clock. Since the audio data of 1fs is 64 bits, the program needs to generate four signals for reading SDRAM.

WM8731 Register Initialization Program reg_config.v Description After the program is powered on, the initialization of the register of the WM8731 chip will be performed through the I2C bus. For the detailed register description of the WM8731 chip, please refer to the datasheet of the chip.

IIC Communication Program i2c_com.v Description

The IIC communication program outputs the external data to the external

IIC bus in time series, thereby realizing the IIC data writing function.

Reset Delay Module reset_delay.v description

This is a method of resetting after software power-on. The purpose is to wait for a period of time after power-on and then configure the register of WM8731.

3. **Button Detection program**

The program will detect if the button KEY1 is pressed or released. If the KEY1 button is pressed, the recording enable signal is high and the SDRAM write address is cleared to 0. If the KEY1 button is detected to be released, the play enable signal is high and the SDRAM read address is cleared.

4. **System Control Module**

The system_ctrl.v program calls the PLL to generate a 100Mhz SDRAM clock. In addition, the system_delay module is called to generate a system-level reset signal

Part 2.4: Download and Test

Connect headphones and audio modules. Be careful not to insert the wrong interface of the earphone. The pink plug of the earphone is inserted into the pink interface of the audio module, and the green plug of the earphone is inserted into the green interface of the audio module.

Compile the project to generate the audio_test.sof file and download the bit file to the FPGA. At this time, we press KEY1 on the development board and say a word to the microphone. After releasing KEY1, we can hear what you said in the earphone.

Part 3: SD card music player routine

Part 3.1: Music Documents

Before this experiment, we need to store several wav format music files in the SD card. Note that the .wav music file format needs to be 16 bits and the sampling frequency is 48 kHz, which is related to the register setting of WM8731. Right click on the .wav file and select Properties to view it.



Figure 3-1: The Properties of .wave file

For music files, users can download music in wav format from the Internet, and then convert it into 16-bit, wav files with a sampling frequency of 48 KHz format, or directly use the wav music files we provide for experiments. Prepared two songs for the experimental labs we provided.

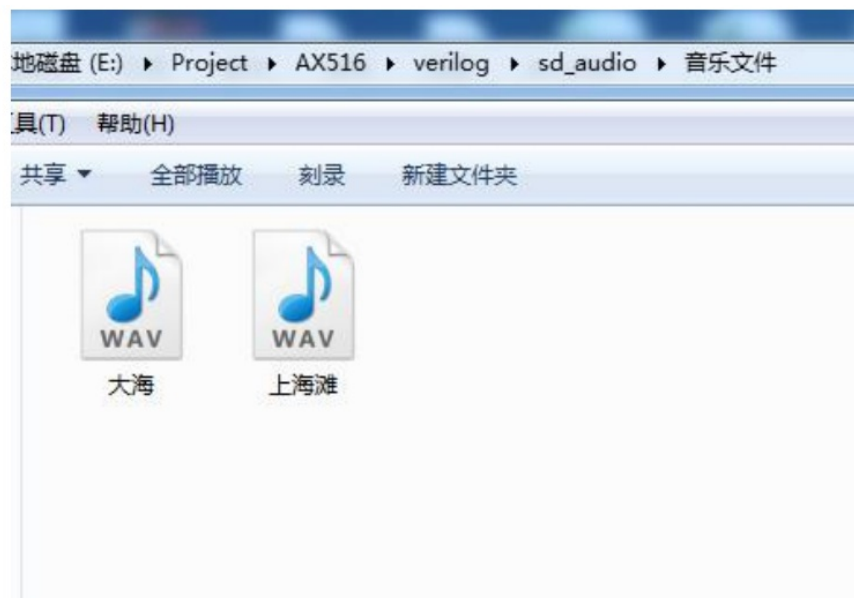


Figure 3-2: Two Songs Prepared for the Experiment

Firstly, use the computer to format the SD card then copy the two songs to the root directory of the SD card, Figure 3-3 detailed the two songs filed in the root directory of the SD card as follows:



Figure 3-3: Two songs filed in the root directory of the SD card

Then check the Sec address of the two songs on the SD card using the win hex tool, which is the starting Sec address when we write the SD card program below:

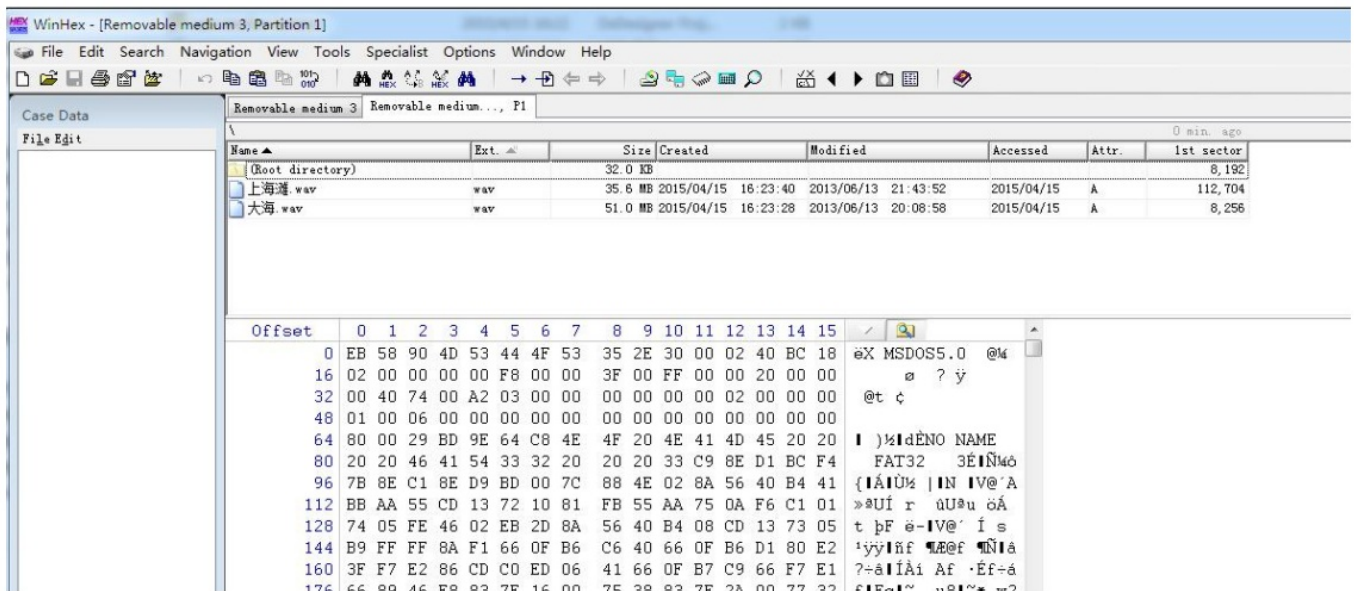


Figure 3-4: Sec address of the root directory of the SD card

In the Win hex window, we can see that the Sec address of the root directory of the SD card is 8192. The Sec address of the song sea is 8256, we can read the data from the address of $8192+8256=16488$ in the program.

Part 3.2: Programming

The whole project sd_audio is composed of a top-level module sd_audio program and several sub-modules (SD card initialization program sd_initial.v, SD card read program sd_read.v, WM8731 audio program mywav.v, ram data storage control program ram_rw_control.v). Figure 3-5 is the project navigator of AX301 FPGA Development board.

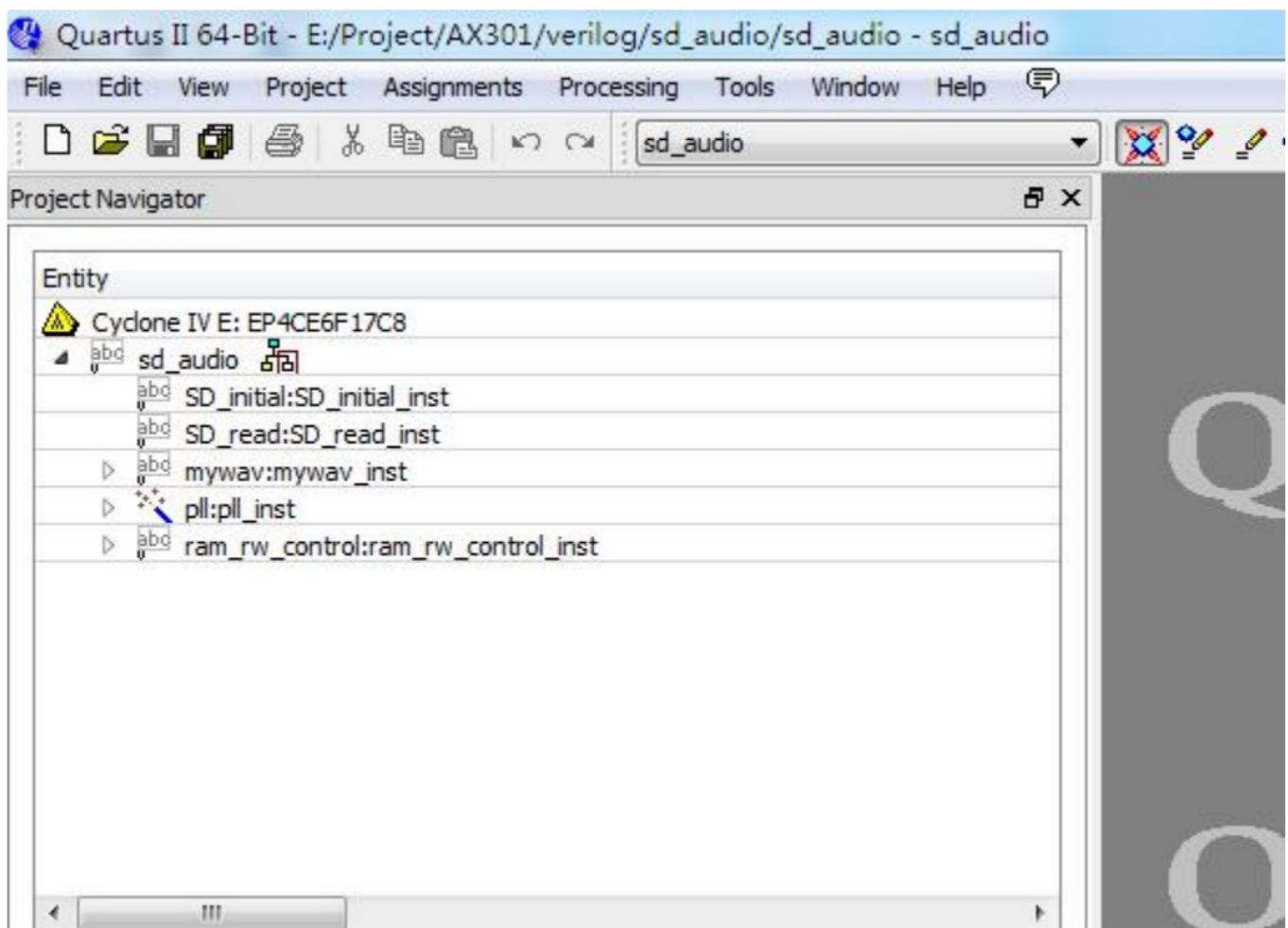


Figure 3-5: The Project Navigator

About the WM8731's I2C register configuration and I2S voice playback, introduced to you in the "Recording and

Playback Routines”, that users can copy these programs directly in this experiment.

In this experiment, the data read by the SD card is stored in the ram array defined by the program. The addresses of the ram, data read and write control completed in the program of raw_porw_control.v. Below briefly introduce and explain the programming of SD card and ram control.

1) SD initialization program design

After the SD card is powered on, it needs to be initialized. For the initialization process and related commands of the SD card, please refer to the protocol standard of SD card 2.0. The initialization process is as follows:

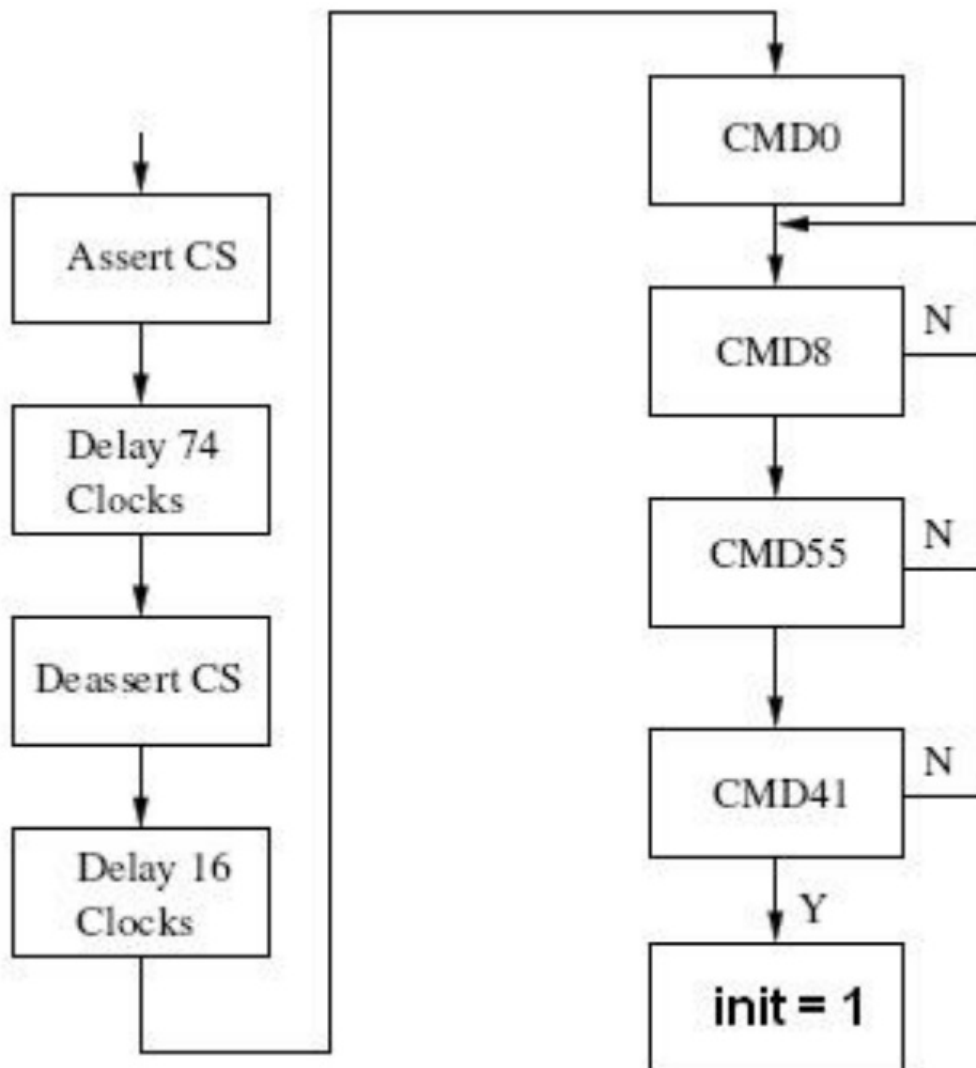


Figure 3-6: SD card initialization process

The specific process description is as follows

- Chip select CS low level to select SD card
- Chip select CS high level release SD card
- Send the CMD0 command to the SD card and confirm if the return is 0x01.
- Send CMD8 command to SD card, CMD8 is a command that is only available in SD2.0. Also check if [bit19:bit16] is returned to 0001 (2.7V-3.6V).
- Send CMD55 command to SD card to confirm whether the return is 0x01CMD55
- Send the ACMD41 command to the SD card to confirm if the return is 0x00.
- If the initialization is successful, the init variable is set to 1. If the Initialization is not successful, resend CMD8, CMD55 and ACMD41 commands

2) SD reading program design

When the SD card is successfully initialized, the data of the image is read into the SDRAM. The program sends a CMD17 single block read command to the SD card, and reads 512 bytes continuously until one image is read.

- The specific process description is as follows
- Send a CMD17 single block read command to the SD card and wait for a response
- Waiting to receive data from the SD card, if the start bit (0) of the data is detected, start receiving data
- Receives 512 bytes of data, converts the serial data received from the SPI into 8-bit data bytes, and outputs one data byte valid signal per byte.
- After receiving 512 bytes of data, determine whether the last data of the image is read.

) ram Control Program Design

The program defines a Myriam register of length 8192 for storing the music data read by the SD card. The program reads and writes the ping-pong structure. When reading the data of the previous 4096 space, the program can write the data of the next 4096 space. Similarly, when writing data in the first 4096 space, the program can read the data program of the next 4096 space. Because the reading speed of the SD card is much faster than the speed of music playback, the ping-pong structure can satisfy the function of reading and simultaneously playing the music data SD card.

After power-on, 4096 data will be read first and stored in the ram register.

After receiving the data read request from the WM8731 audio program mywav.v, the program sends the 2-byte data in the ram register to the mywav.v program, and the read address of the ram increases. In addition, the program judges the read address. When the address of the read ram is 2 or 4096, the SD card is read and 4096 data is read.

The following two lines of music files in the code are the starting sector address of the SD card and the length of the Sector.


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

Figure 3-7: Two lines of music files in the code

Part 3.3: Download and test

Insert the SD card with the music file into the FPGA development board and plug the audio module AN831, then connect the earphone to the green audio output interface of the audio module. After download the .sofa file in the Quartus software, We can hear the wonderful music in the earphones.



<div>Audio Module AN831</div> <div>User Manual</div> <div></div>	<div>ALINX AX415 Audio Module [pdf] User Manual</div> <div>AX415 Audio Module, AX415, Audio Module, Module</div>
---	--

References

-  [ALINX](#)