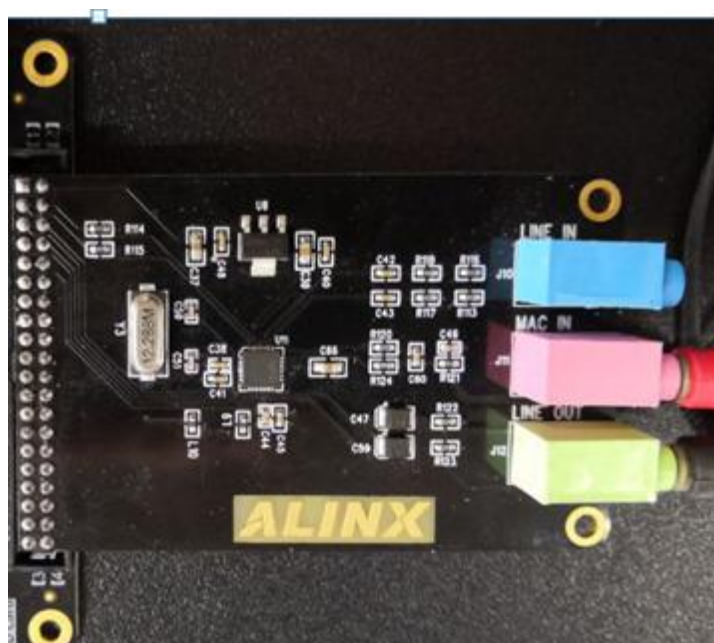


# 录音与播放例程

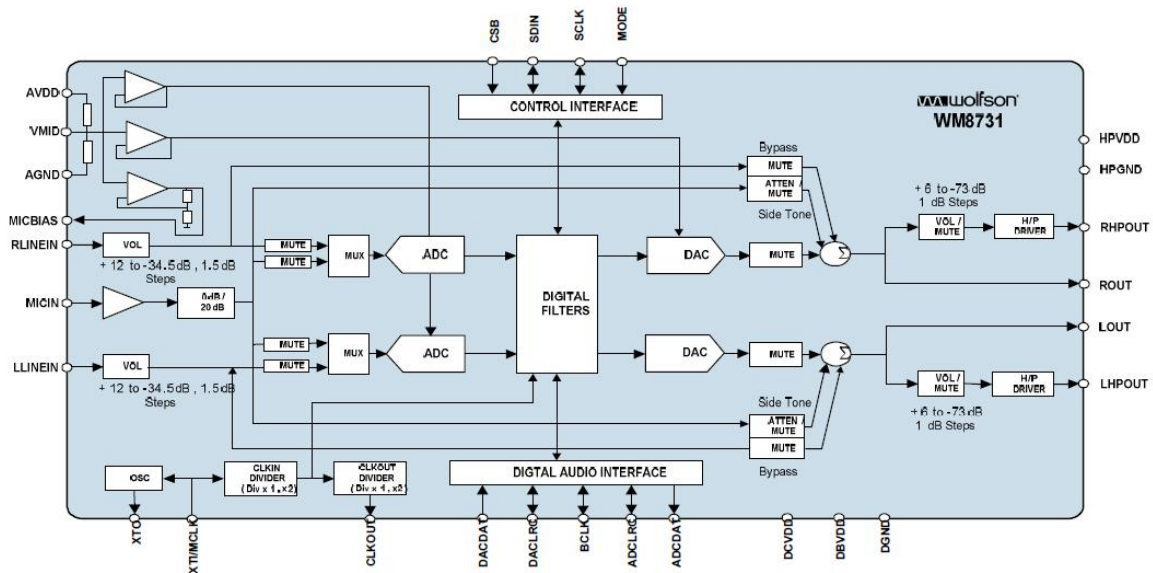
## 1 实验简介

本实验的录音和播放实验因为开发板上没有音频部分的电路，需要外接一个芯驿电子的 AUDIO 音频模块 AN831。



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 DDR3 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。

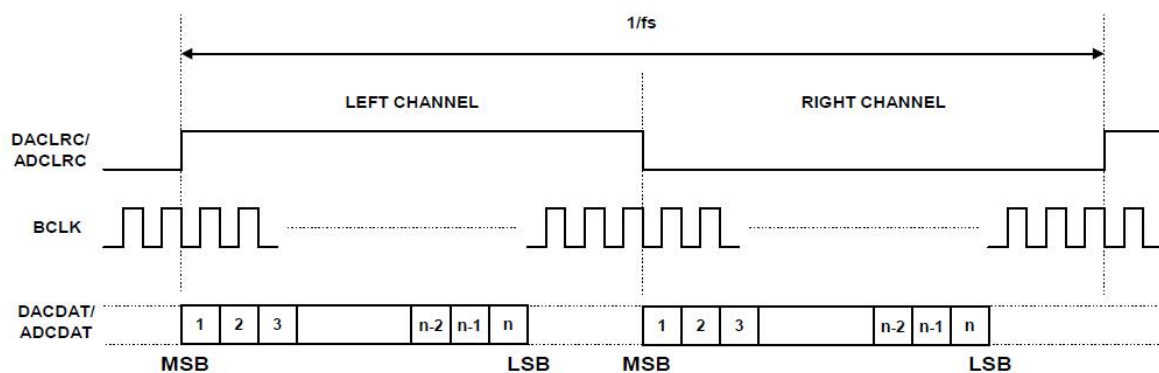




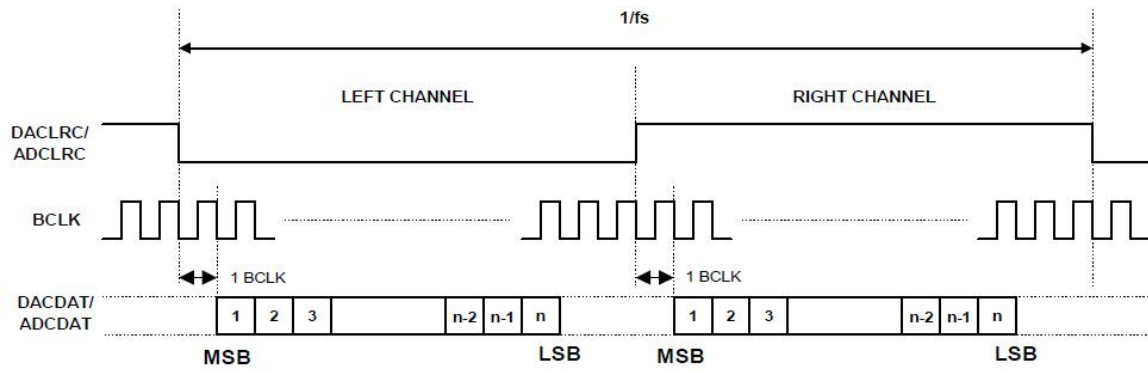
在本系统中 FPGA 和 WM8731 的控制和数据通信将用到 I2C 和数字音频总线接口。FPGA 通过 I2C 接口配置 WM8731 的寄存器，通过 I2S 总线接口来进行音频数据的通信。关于 I2C 接口，其他实验中已经有讲解，下面我们主要来了解数字音频接口。

数字音频接口可提供 4 种模式：

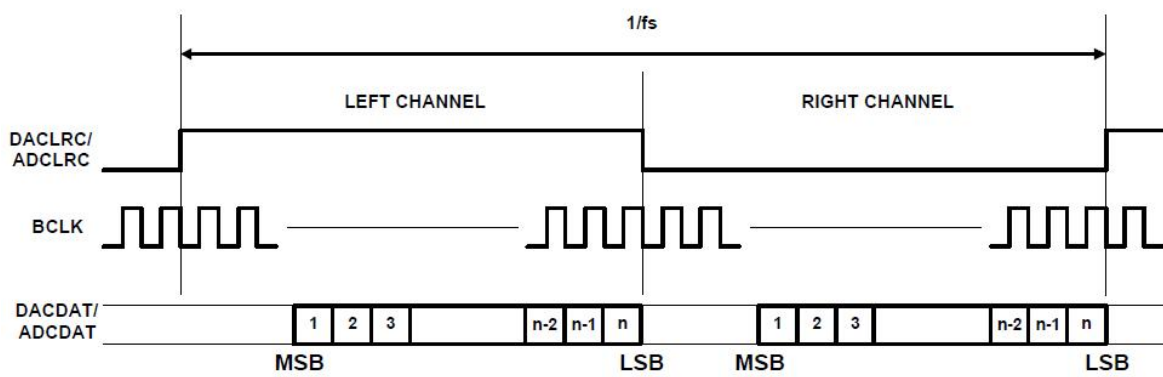
- Right justified
- Left justified
- I2S
- DSP mode



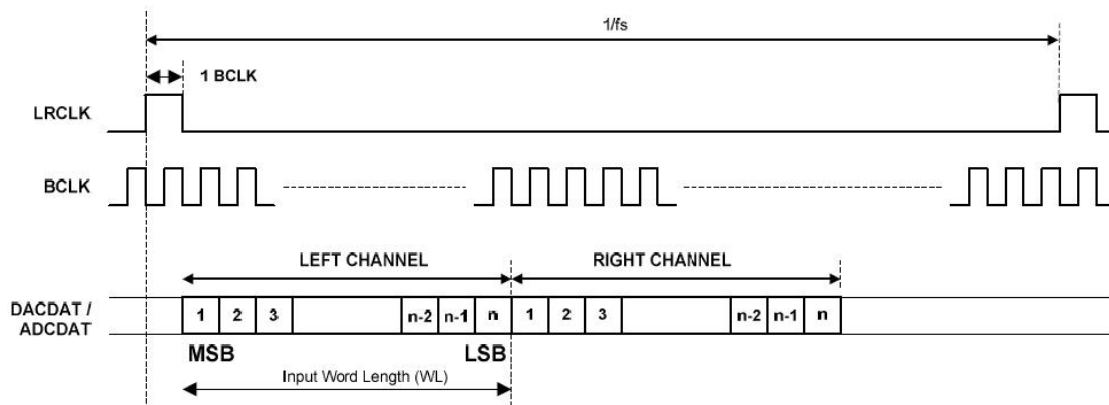
Left justified 模式



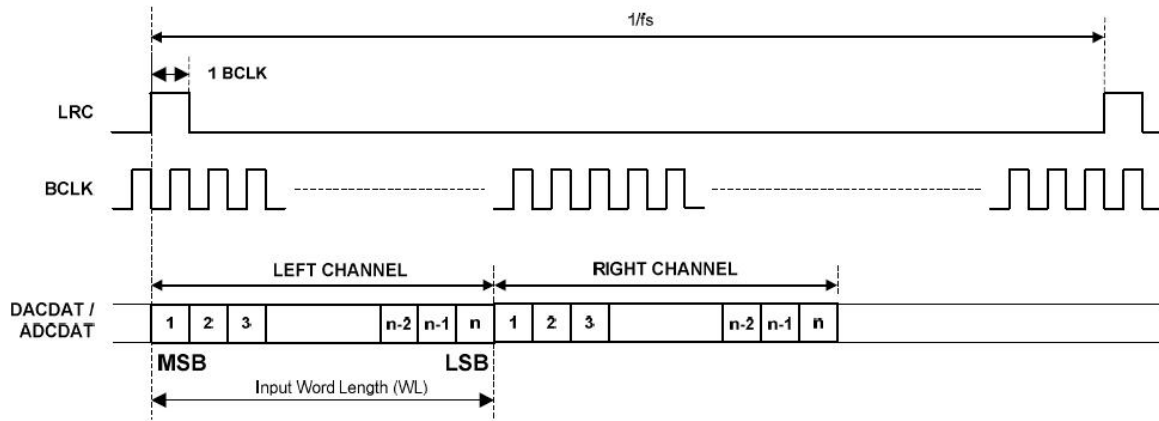
I2S 模式



Right justified 模式



DSP/PCM 模式 (MODE A)



DSP/PCM 模式 (MODE B)

本实验选择 Right justified 模式。

| REGISTER ADDRESS                          | BIT | LABEL       | DEFAULT | DESCRIPTION   |
|---|-----|-------------|---------|---|
| 0000111<br>Digital Audio Interface Format | 1:0 | FORMAT[1:0] | 10      | Audio Data Format Select<br>11 = DSP Mode, frame sync + 2 data packed words<br>10 = I <sup>2</sup> S Format, MSB-First left-1 justified<br>01 = MSB-First, left justified<br>00 = MSB-First, right justified  |
|   | 3:2 | IWL[1:0]    | 10      | Input Audio Data Bit Length Select<br>11 = 32 bits<br>10 = 24 bits<br>01 = 20 bits<br>00 = 16 bits  |
|   | 4   | LRP         | 0       | DACLRC phase control (in left, right or I <sup>2</sup> S modes)<br>1 = Right Channel DAC data when DACLRC high<br>0 = Right Channel DAC data when DACLRC low<br>(opposite phasing in I <sup>2</sup> S mode)<br>or<br>DSP mode A/B select (in DSP mode only)<br>1 = MSB is available on 2nd BCLK rising edge after DACLRC rising edge<br>0 = MSB is available on 1st BCLK rising edge after DACLRC rising edge |
|   | 5   | LRSWAP      | 0       | DAC Left Right Clock Swap<br>1 = Right Channel DAC Data Left<br>0 = Right Channel DAC Data Right  |
|   | 6   | MS          | 0       | Master Slave Mode Control<br>1 = Enable Master Mode<br>0 = Enable Slave Mode  |
|   | 7   | BCLKINV     | 0       | Bit Clock Invert<br>1 = Invert BCLK<br>0 = Don't invert BCLK  |

### 3 程序设计

本实验的功能是程序检测按键 KEY2 是否按下，如果检测到 KEY2 按下了，开始录音,录音的最长时间为 20 秒；录音结束后，开始播放刚才录下的音频。本程序设计包含四大部分：DDR3 的读写控制程序，音频录音和播放，按键检测和系统控制。

lut\_wm8731 模块将寄存器配置地址和配置信息通过查找表的形式通过 I2C 总线写入音频模块中，具体的请参考例程代码和 wm9731 的芯片数据手册。

audio\_rx 接收模块，接收从麦克风输入的语音信号，完成左右声道的音频接收，将串行数据转换成并行数据。通过“Right justified”模式的时序图可以看到接收语音信号时在 LRC 信号为高电平，且 BCLK 信号的上升沿时左声道接收数据并完成串行信号转换成并行信号的过程。在 LRC 信号为低电平，且 BCLK 信号的上升沿时右声道接收数据并完成串行信号转换成并行信号的过程。

| 信号名称       | 方向  | 说明            |
|------------|-----|---------------|
| clk        | in  | 时钟输入          |
| rst        | in  | 异步复位输入，高复位    |
| sck_bclk   | in  | 数字音频接口 bit 时钟 |
| ws_lrc     | in  | ADC 采样时钟      |
| sdata      | in  | 音频数字接口串行数据输入  |
| left_data  | out | 左声道数据         |
| right_data | out | 右声道数据         |
| data_valid | out | 音频数据有效        |

audio\_rx 音频接收模块端口

audio\_tx 是音频发送模块，完成左右声道音频数据的串行化。同样通过时序图可以看到，语音信号完成模数，数模转换从 DDR3 输出后进入发送模块，在 LRC 信号上升沿，且即将跳变为高电平时接收左声道数据，BCLK 信号下降沿时完成缓存将并行信号转换为串行信号的过程；在 LRC 信号上升沿且即将跳变为低电平时接收右声道数据，在 BCLK 信号下降沿时完成缓存将并行信号转换为串行信号的过程。

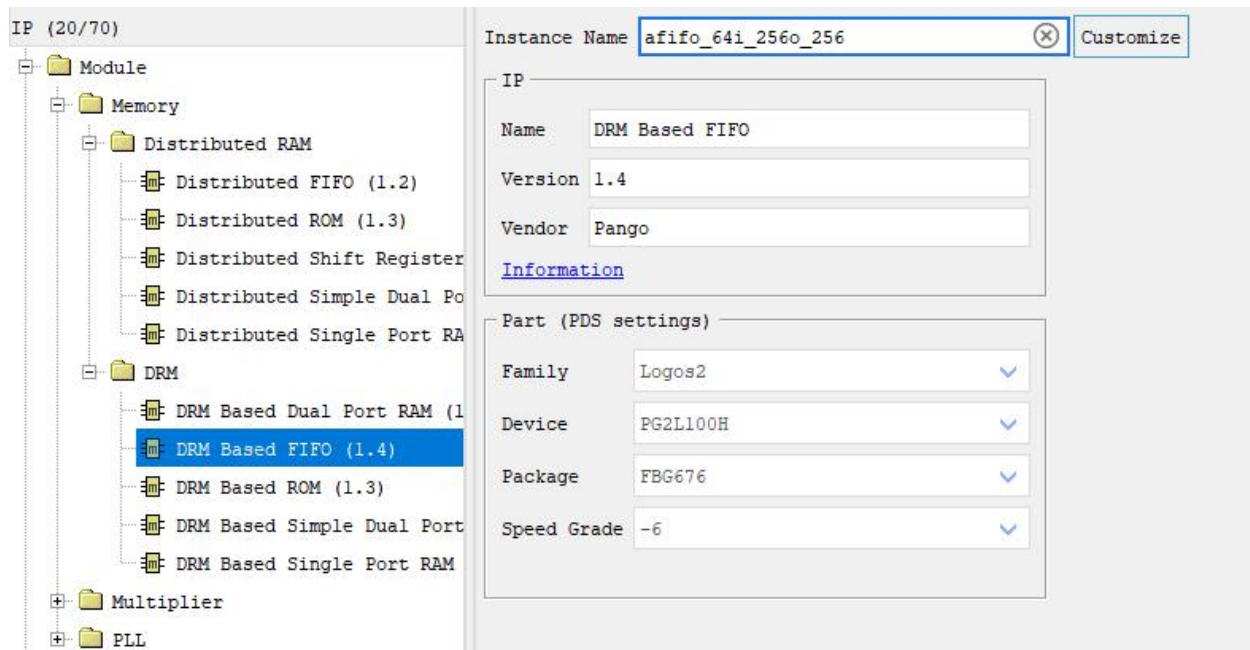
| 信号名称         | 方向  | 说明                |
|--------------|-----|-------------------|
| clk          | in  | 时钟输入              |
| rst          | in  | 异步复位输入，高复位        |
| sck_bclk     | in  | 数字音频接口 bit 时钟     |
| ws_lrc       | in  | ADC 采样时钟          |
| sdata        | out | 音频数字接口串行数据输入      |
| left_data    | in  | 左声道数据             |
| right_data   | in  | 右声道数据             |
| read_data_en | out | 音频数据读取，提前一个采样周期读取 |

audio\_tx 音频发送模块端口



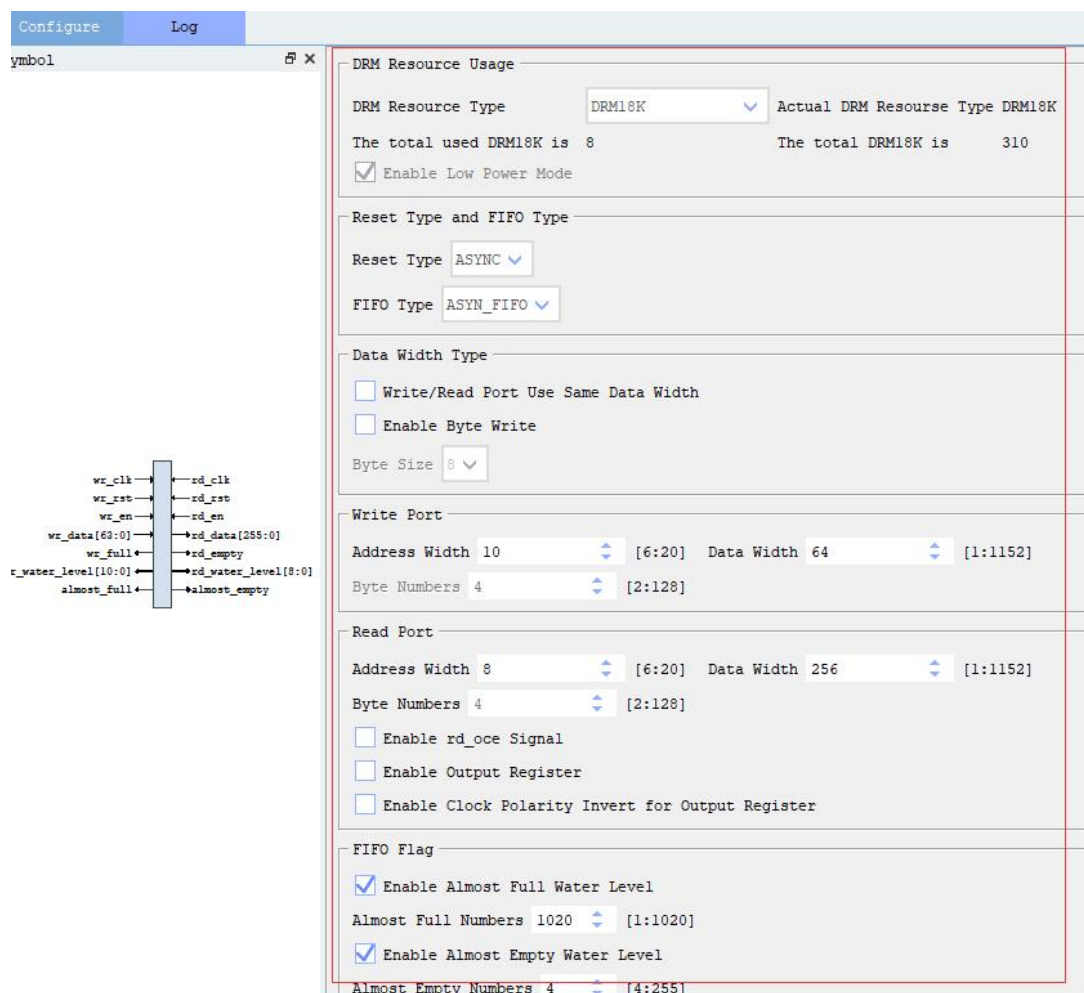
在模块 'frame\_read\_write' 中使用到了 FIFO 的 IP core，通过两个 FIFO 分别作为 DDR3 控制器的读写接口，避免复杂的 DDR3 时序。因为时钟速率的不同，所使用的是异步 FIFO。以 write\_buf 写入模块的 FIFO 为例在 vivado 中 FIFO IP core 设置如下：

1. 打开 Tools 菜单下的 "IP Compiler" ， 在弹出的界面下进行如下设置，设置完成后点击 " Customize" ：



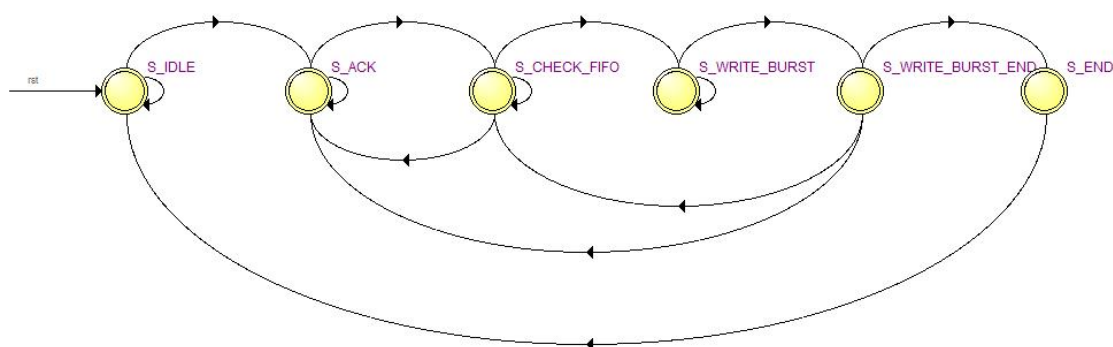
2. 在弹出的界面下进行如下设置，设置完成后点击保存后 Generate 即可生成 IP：





- 接着在模块中直接例化 FIFO 的端口名就可以使用 FIFO 了，在 read\_buf 中也是同样的操作步骤。具体的参数设置和端口例化的信号连接参考例程。

frame\_fifo\_write 模块完成 FIFO 数据到外部存储器的写入，如果 FIFO 接口是异步 FIFO，可以完成写数据的跨时钟域转换。状态机转换图如下，收到写数据请求后进入应答状态 “S\_ACK”，如果写请求撤销，则进入检测 FIFO 空间大小状态 “S\_CHECK\_FIFO”，检查 FIFO 内数据是否够一次突发写，如果有足够多的数据，进入突发写存储器状态 “S\_WRITE\_BURST”，突发写完成后进入 “S\_WRITE\_BURST\_END” 状态。



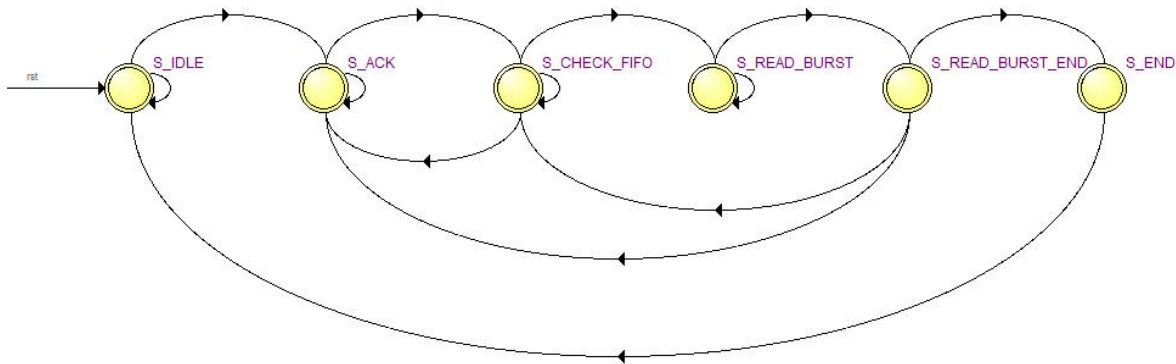
frame\_fifo\_write 模块状态机

| 信号名称              | 方向  | 说明   |
|-------------------|-----|--|
| mem_clk           | in  | 外部存储器用户时钟输入                                |
| rst               | in  | 异步复位输入，高复位                                 |
| wr_burst_req      | out | 对接存储器控制器，写请求                               |
| wr_burst_len      | out | 对接存储器控制器，写请求长度                             |
| wr_burst_addr     | out | 对接存储器控制器，写请求基地址                            |
| wr_burst_data_req | in  | 对接存储器控制器，写请求数据索取，提前一个时钟周期发出，用于连接 FIFO 的读数据 |
| wr_burst_finish   | in  | 对接存储器控制器，写请求完整                             |
| write_req         | in  | 一帧（大量数据）写开始，收到应答后必须撤销请求，新的请求会中断正在进行的请求     |
| write_req_ack     | out | 一帧（大量数据）写应答                                |
| write_finish      | out | 一帧（大量数据）完成                                 |
| write_addr_0      | in  | 一帧（大量数据）写基地址 0                             |
| write_addr_1      | in  | 一帧（大量数据）写基地址 1                             |
| write_addr_2      | in  | 一帧（大量数据）写基地址 2                             |
| write_addr_3      | in  | 一帧（大量数据）写基地址 3                             |
| write_addr_index  | in  | 一帧（大量数据）写基地址选择，<br><br>0: write_addr_0     |

|           |     |   |
|-----------|-----|---|
|           |     | 1: write_addr_1<br><br>2: write_addr_2<br><br>3: write_addr_3 |
| write_len | in  | 一帧（大量数据）写长度   |
| fifo_aclr | out | 在收到写请求后，模块会清空 FIFO  |
| rdusedw   | in  | FIFO 读端的数据使用量   |

frame\_fifo\_write 模块端口

frame\_fifo\_read 读模块完成从外部存储器读取数据，然后写到 FIFO，如果使用异步 FIFO 可以完成数据从存储器时钟域到其他时钟域的转换。状态机转换图如下图所示，收到读请求以后进入应答状态 “S\_ACK”，等待读请求撤销后应答，进入 FIFO 深度检测状态 “S\_CHECK\_FIFO”，如果 FIFO 空间足够一次突发读，进入突发读状态 “S\_READ\_BURST”，突发读结束后进入 “S\_READ\_BURST\_END”。



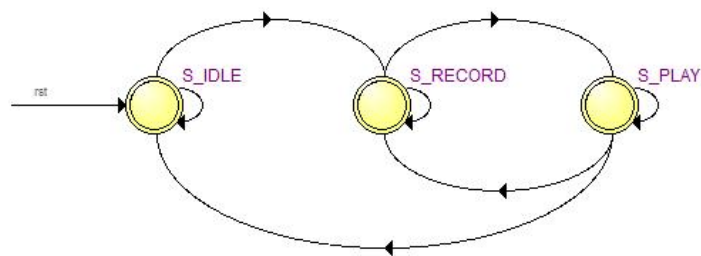
frame\_fifo\_read 模块状态机

| 信号名称                | 方向  | 说明               |
|---------------------|-----|------------------|
| mem_clk             | in  | 外部存储器用户时钟输入      |
| rst                 | in  | 异步复位输入，高复位       |
| rd_burst_req        | out | 对接存储器控制器，读请求     |
| rd_burst_len        | out | 对接存储器控制器，读请求长度   |
| rd_burst_addr       | out | 对接存储器控制器，读请求基地址  |
| rd_burst_data_valid | in  | 对接存储器控制器，读请求数据有效 |

|                        |     |  |
|------------------------|-----|--|
| <b>rd_burst_finish</b> | in  | 对接存储器控制器，读请求完全   |
| <b>read_req</b>        | in  | 一帧数据读开始  |
| <b>read_req_ack</b>    | out | 一帧数据读应答  |
| <b>read_finish</b>     | out | 一帧数据读完成  |
| <b>read_addr_0</b>     | in  | 一帧数据读基地址 0   |
| <b>read_addr_1</b>     | in  | 一帧数据读基地址 1   |
| <b>read_addr_2</b>     | in  | 一帧数据读基地址 2   |
| <b>read_addr_3</b>     | in  | 一帧数据读基地址 3   |
| <b>read_addr_index</b> | in  | 一帧数据读基地址选择<br><br>0: read_addr_0<br><br>1: read_addr_1<br><br>2: read_addr_2<br><br>3: read_addr_3 |
| <b>read_len</b>        | in  | 帧数据读长度   |
| <b>fifo_aclr</b>       | out | 外部 FIFO 异步复位   |
| <b>wrusedw</b>         | in  | FIFO 写端使用空间大小  |

frame\_fifo\_read 模块端口

audio\_key 模块主要完成录音播放的按键控制，状态转换如下，当按键按下后进入录音状态，当按键松开时进入播放状态。



audio\_key 模块状态转换图

| 信号名称          | 方向  | 说明         |
|---------------|-----|------------|
| clk           | in  | 时钟输入       |
| rst           | in  | 异步复位输入，高复位 |
| key           | in  | 按键输入       |
| record        | out | 录音状态指示     |
| play          | out | 播放状态指示     |
| write_req     | out | 写数据开始      |
| write_req_ack | in  | 写数据应答      |
| read_req      | out | 读数据开始      |
| read_req_ack  | in  | 读数据应答      |

frame\_read\_write 模块完成音频帧数据读写的封装，这里使用了异步 FIFO 来解决跨时钟问题，例如 FIFO 输入宽度和输出宽度的不同来完成数据位宽的转换。

## 4 实验现象

开发板首先将音频模块插入 J11 扩展口，如下图所示，同时插入麦克风和耳机，然后下载实验程序，按下 KEY2 不放，进行录音，松开按键后可通过耳机回放录音。

