數位電路實驗

Lab 3 report

Audio Recorder

team 06

王珽 B05901016

林承德 B05901064

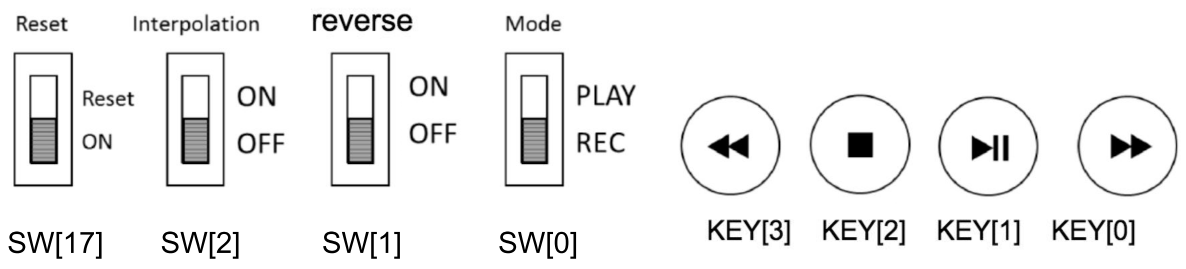
戴慕潔 B05901185

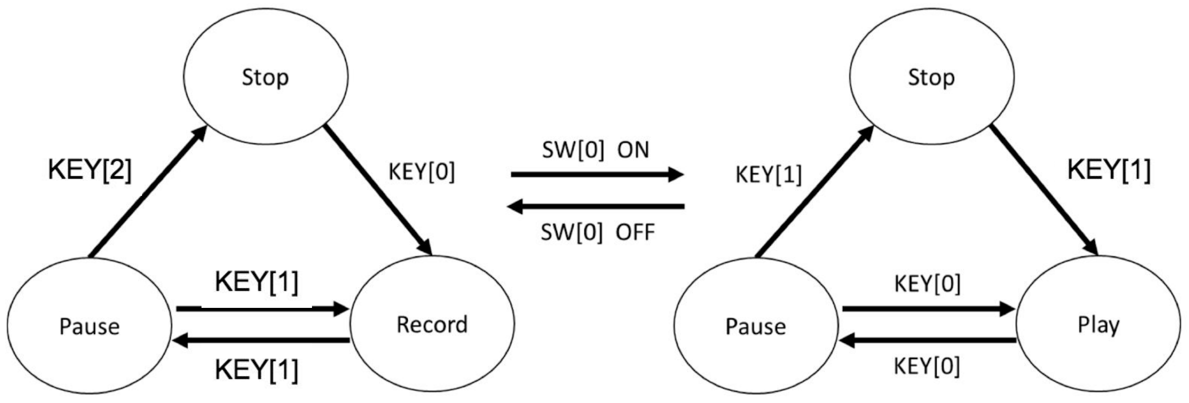
- **github**
  https://github.com/whoami90506/dclab_lab3.git
- **Introduction**
  In this lab, we are going to implement an audio recorder that is capable of record, play, pause and stop. Beside these basic functions, the audio recorder should be able to play at different speed as well as provide a user-friendly interface. In addition, we will let the red LED lights indicate the progress of recording or playing and the green LED lights indicate the volume as some extra bonus.
- **Hardware**: Altera DE2-115 FPGA board
- **Software**: Quartus II
- **HDL (Hardware Description Language)**: System Verilog

Part I: User Manual

- Specification
  - Sampling Rate: 32 kHz
  - Sampling Length: 16 bits
  - Sound Reproduction: Monaural
  - Longest Recording Time: 32 seconds
- Usage
  - Connect your microphone and speaker (or earphone) to the FPGA board.
  - Record
    i. Switch SW[0] downward to the RECORD mode.
    ii. Press KEY[1] to start recording.
    iii. You may press KEY[1] multiple times during recording to pause and press again to continue.
    iv. When recording, the green LED lights will show your relative volume, and if the light go toward, it means that your volume is too high.
    v. The HEX6 and HEX7 of SevenHexDecoder represent the seconds that are recorded.
    vi. To stop the recording process, press KEY[1] first to pause and then press KEY[2] to stop.

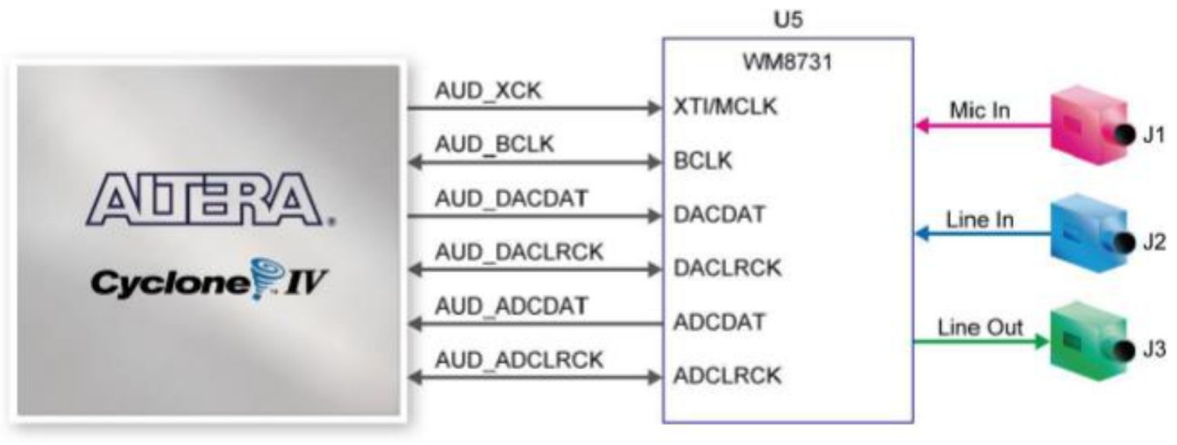
- Play
    1. Switch SW[0] upward to the PLAY mode.
    2. Press KEY[1] to start playing the recorded audio.
    3. You may press KEY[1] multiple times during playing to pause and press again to continue.
    4. You can speed up the audio by pressing KEY[0] or slow down by pressing KEY[3]. The audio recorder supports 2x up to 8x and 1/2x down to 1/8x speed.
    5. In slow-forward replaying, you can enable linear interpolation by switching SW[1] upward to ON.
    6. The HEX6 and HEX7 of SevenHexDecoder represent the seconds that are played.
    7. If you want to reverse the sound, switch SW[1] on.
    8. To stop the playing process, press KEY[1] first to pause and press KEY[2] to stop.



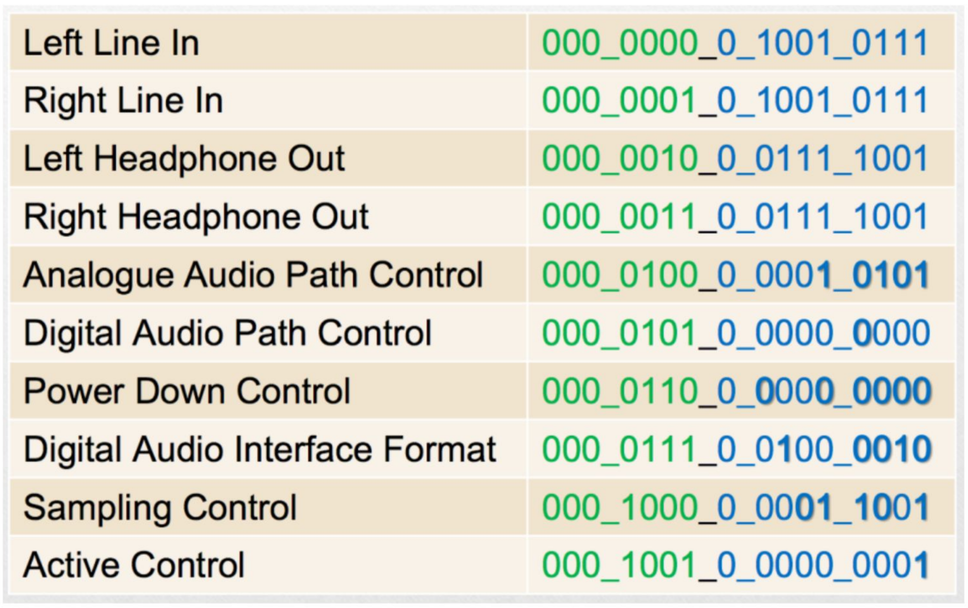


Part II: Tutorial

- Introduction of Protocols and Tools
    - WM8731 Audio CODEC
        - Schematic Diagram:



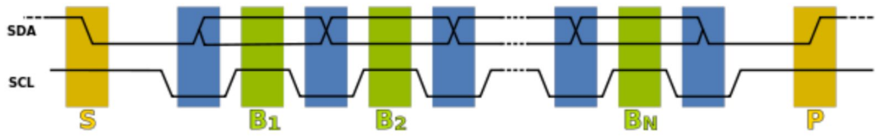
        - Usage:
          First, we need to set up the WM8731 audio CODEC by changing its register map, and to do so, we must pass configuration to it with the I2C protocol. The initialization configuration for this lab are as follows.



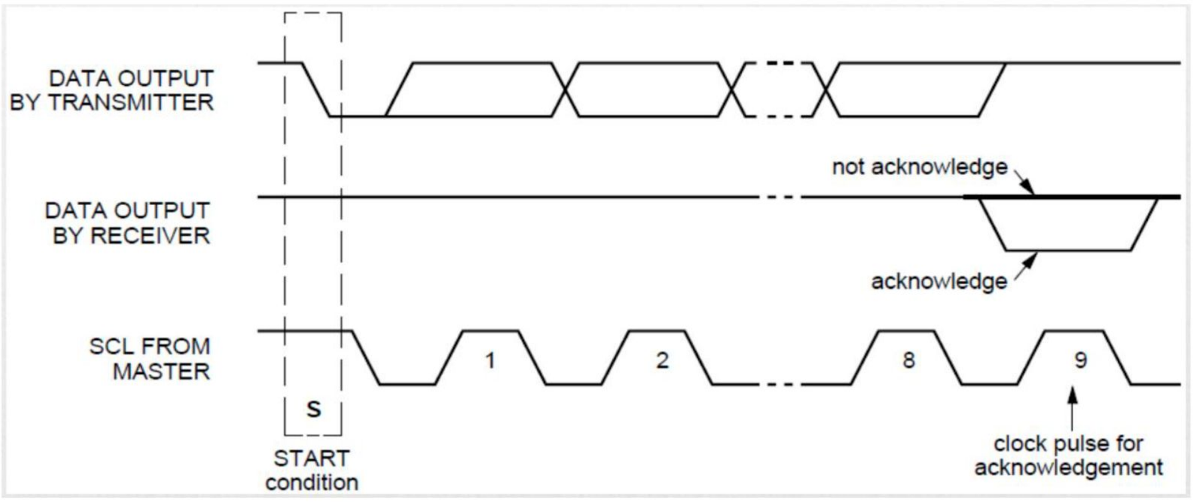
Left Line In	000_0000_0_1001_0111
Right Line In	000_0001_0_1001_0111
Left Headphone Out	000_0010_0_0111_1001
Right Headphone Out	000_0011_0_0111_1001
Analogue Audio Path Control	000_0100_0_0001_0101
Digital Audio Path Control	000_0101_0_0000_0000
Power Down Control	000_0110_0_0000_0000
Digital Audio Interface Format	000_0111_0_0100_0010
Sampling Control	000_1000_0_0001_1001
Active Control	000_1001_0_0000_0001

After initializing, WM8731 will be activated as DAC and ADC between audio signal and the FPGA. The sample rate, or the frequency of DACLRCK/ ADCLRCK, is set to 32kHz. Each DACLRCK/ADCLRCK clock cycle contains one sample of the audio signal, which is 16 bits long in our case. The serial data transmission format follows the rules under I2S mode.

- I^{2}C Protocol
    - The picture below is a typical waveform under I^{2}C protocol, and we shall use it as an example to explain how I^{2}C protocol works.

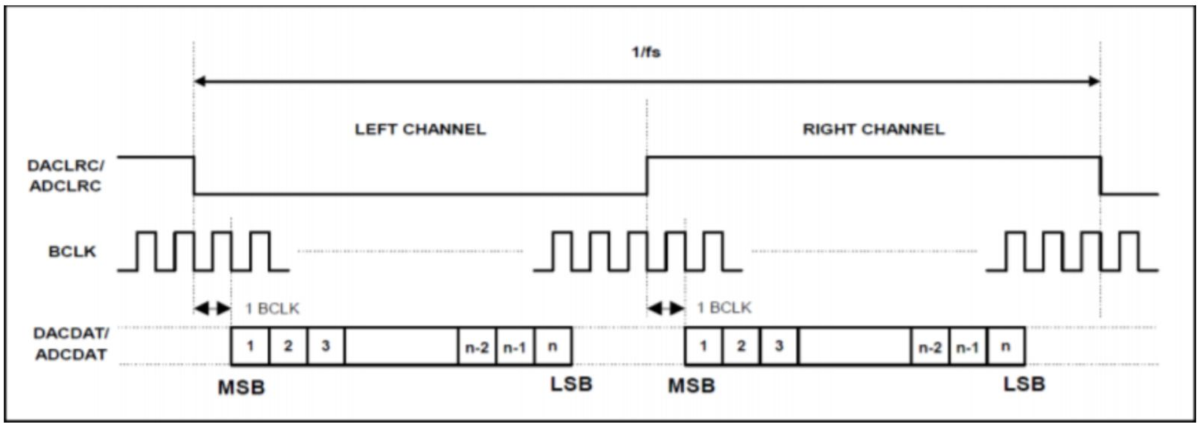


1. The two signals of I^{2}C protocol, namely SDA and SCL, are preset to high.
2. To start transmission, SDA is pulled low while SCL is high. (refer to the yellow region S)
3. To stop transmission, SDA is pulled high while SCL is high. (refer to the yellow region P)
4. During transmission, SDA sets the data bit to be transferred when SCL is low. (refer to the blue region) While SCL is high, SDA remains unchanged and the data bit is transmitted. (refer to the green region)
5. After transmitting every 8 bits is one direction, an “acknowledgement” bit ACK is transmitted in the opposite direction. If ACK is high, it indicates that the transmission failed; on the other hand, if ACK is low, the transmission is successful. (refer to the figure below)

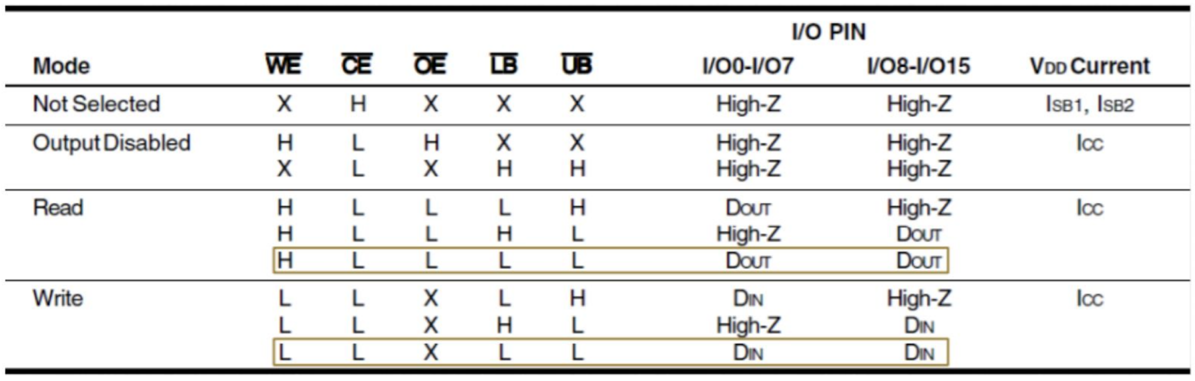


- I^{2}S Protocol

I2S mode is one of the audio interfaces offered by WM8731. In a single DACLRCK/ADCLRCK clock, DACLRC/ADCLRC low means the left channel while high means the right channel. When DACLRCK/ADCLRCK changes value, DACDAT/ADCDAT will start to record data after one BCLK clock. The number of bits recorded is 16 in our case, and transmission starts from the MSB to the LSB.



- SRAM Communication:
  The SRAM on DE2_115 has roughly 2MB memory capacity and is capable of storing 1024K words of 16 bits. There are seven signals associated with the SRAM:
    - SRAM_ADDR[19:0]: The read/write address of the SRAM.
    - SRAM_DQ[15:0]: The 16 bits data to be read or stored in the SRAM. Note that this is a inout port, which means that one must beware of the control of this signal to avoid multiple
    - driver of signal.
    - SRAM_OE SRAM: Output Enable
    - SRAM_WE SRAM: Write Enable
    - SRAM_CE SRAM: Chip Enable
    - SRAM_LBSRAM:LowerByteControl
    - SRAM_UB SRAM: Upper Byte Control
      The control signals of the read/write operation are listed as below:



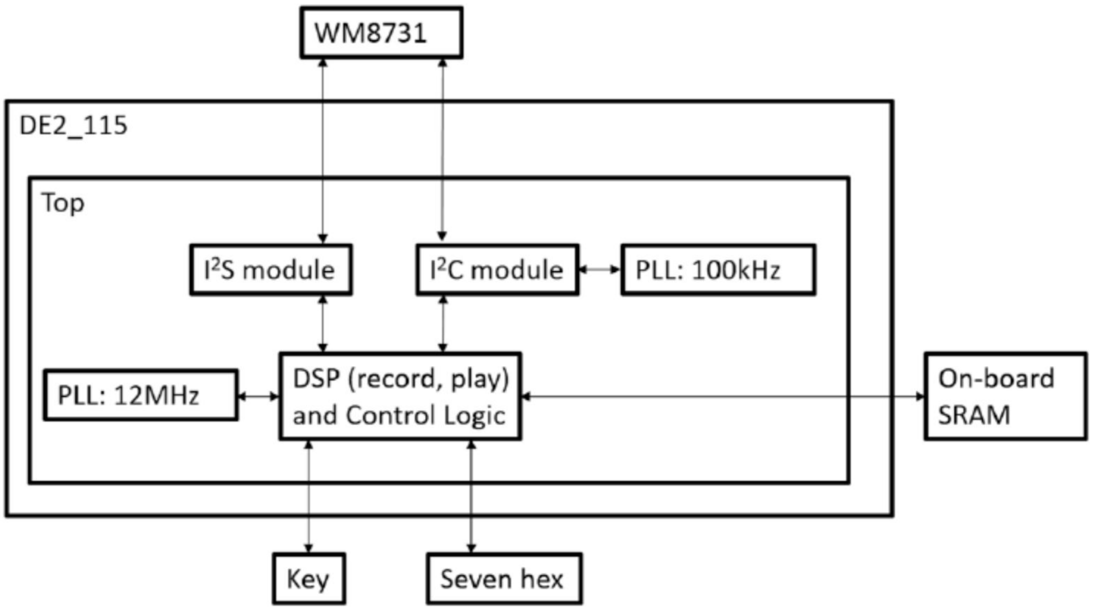
						I/O PIN		
Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	$\overline{LB}$	$\overline{UB}$	I/O0-I/O7	I/O8-I/O15	V_{DD} Current
Not Selected	X	H	X	X	X	High-Z	High-Z	I_{SB1}, I_{SB2}
Output Disabled	H	L	H	X	X	High-Z	High-Z	I_{CC}
	X	L	X	H	H	High-Z	High-Z	
Read	H	L	L	L	H	D_{OUT}	High-Z	I_{CC}
	H	L	L	H	L	High-Z	D_{OUT}	
	H	L	L	L	L	D_{OUT}	D_{OUT}	
Write	L	L	X	L	H	D_{IN}	High-Z	I_{CC}
	L	L	X	H	L	High-Z	D_{IN}	
	L	L	X	L	L	D_{IN}	D_{IN}	

Another important point to notice is that although the WM8731 audio CODEC transmits 2's complement formatted binary data in a serial way, SRAM transmits data in a parallel way. In other words, the 16-bit data transmitted from WM8731 can be written into SRAM in a single clock cycle.

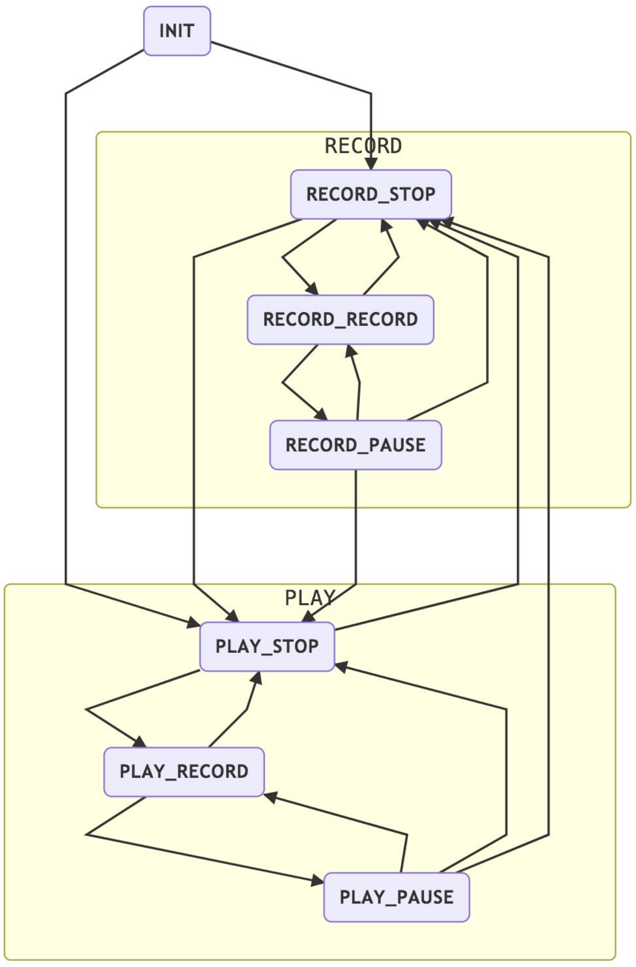
- ALTPLL
  ALTPLL is a clock rate conversion tool provided by Quartus II. In this lab, we use PLL to convert the original clock rate from 50 MHz to 12 MHz and 100 kHz. The 12MHz clock is then fed to the AUD_XCK port and the 100kHz clock is used to initialize WM8731 through I2C protocol. For other modules, we use the clock signals generated by WM8731, which is set to 12 MHz and 32 kHz respectively, as the primary clock signals.

Part III: Design

- Overall Structure:



- Finite State Machine



- I²C

  I²C mainly initilize the WM8731 audio chip after the reset signal from users. This module use the clock with 100kHz to send several instructors to the audio chip by the I²C protocol including proper shake hand. After finishing initializing the audio chip, this module will tell the top module in order to start functions users need.

- Top

  It mainly receives user input signal and declare current state (and speed) for all the other submodule.

- I²S

  There are two submodule in I²S, named as ADC and DAC. ADC tranform the serial audio data from WM8731 chip to parallel data. Since there is only one channel in this project, we just save the left channel. DAC tranform the parallel audio data to serial data and send it to WM7831. Since there is only one channel in this project, we send the same data to both left and right channels for WM8731.

I^{2}S proccesses the data transmission between ADC, DAC and other module such as SRAM and DSP. If ADC/DAC sends/requests audio datas, it will decide whether sends to SRAM/ requests from DSP based on the current state from top module.

- DSP

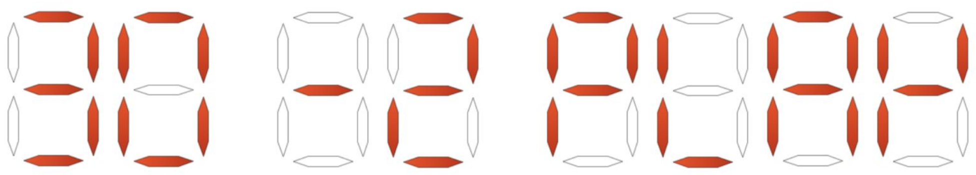
DSP mainly interacts with I^{2}S and SRAM, using the input wire I2S_request_data and data_valid and the output wire request_data and valid. When I^{2}S requests data, DSP will get the data from SRAM to the input wire data_in, and DSP will manage the data_in with the play_speed, which is given by Top module. The play_speed has 15 modes(normal, 2~8 times speed, ½ ~ ⅛ time speed). In the seven accelerating modes, we transfer the value to data_out with once every two~eight data_input. As to the slow-down mode, we have zero-slot（零階) and one-slot（一階）modes. In zero-slot mode, we repeat one data_in value to two~eight times and give values to data_out. On the other hand, in one-slot mode, we record two consequential data_valid, use the linear way to insert one~seven point in the two data_input point, and give values to data_output. After a great deal of manipulation, we can gain the appropriate data_out with DSP and give it to I^{2}S.

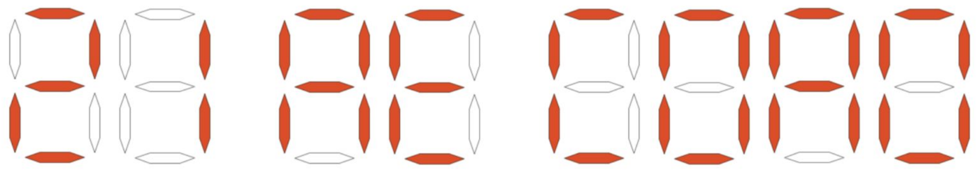
- SRAM

SRAM directly communicates with the SRAM on FPGA board(i.e. Sram Controller). It records the address which the recorder is playing or recording at and which the audio ends at in order to give right playing audio data to DSP and store recording audio data from I2S to proper address. Moreover, SRAM need to tell module top that sram is full of audio data when recording and that the users reach the end of data when playing, and the top will change the state to STOP.

Part III: Bonus

- Seven Segment Display
    1. Timer
       Two seven segment displays represent playing or recording time in seconds. As mentioned above, SRAM stores audio digital signals at address range from 0 to 220, we therefore split 0 to 220 into 32 intervals (since the audio recorder records for at most 32 seconds), and by identifying the current reading/writing address interval of SRAM, we are able to show the correct timer of audio recorder by seven segment display. For example, when SRAM manipulating data at address 0001_1000_0000_0000_0001, seven segment display will show 2 digits number ’04’. Note that we are able to check only most significant 5 bits to identify the address interval.
    2. State Machine
       For users to know the current state of the recorder, we utilize the rest of seven segment displays to show the current state of the machine. There are 6 states in total, including init, idle, record, stop, play, and pause. Further more, since user may play the recorder at different speeds, we also show the playing speed while the recorder is playing. Two examples of our seven segment displays are shown on the right.


Playing at Speed 1/2 to 30 seconds


Recording to 21 seconds

- reverse

  When playing audio, the user can select to play normally or play reversely. In addition, user can play reversely with several speed as normal playing(i.e. ⅛~8 times). To implement this functions, the SRAM will give DSP data in descending order address in sram if users intend to play reversely.

- LED volume indicator

  When recording, the red and green LEDs will show the current volume of the users' voice. The loader the user speaks, the more LEDs are light. The specific green LED between two seven segment displays indicates whether the users speak so loud that the data is overflow or not.