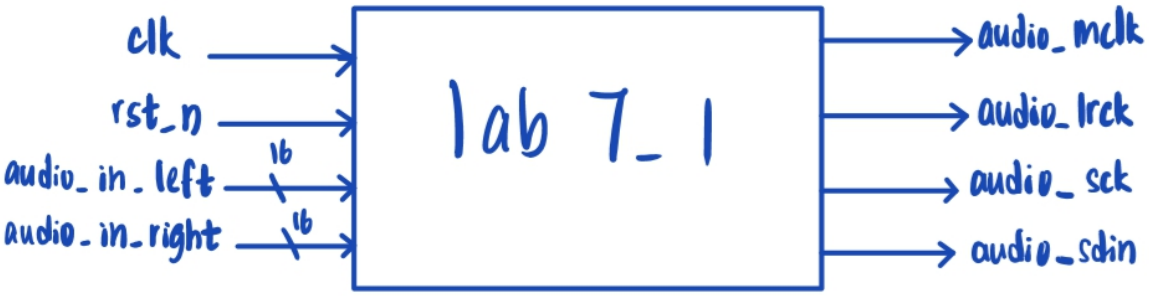
Lab 7 Report

Design Specification

Lab7_1:

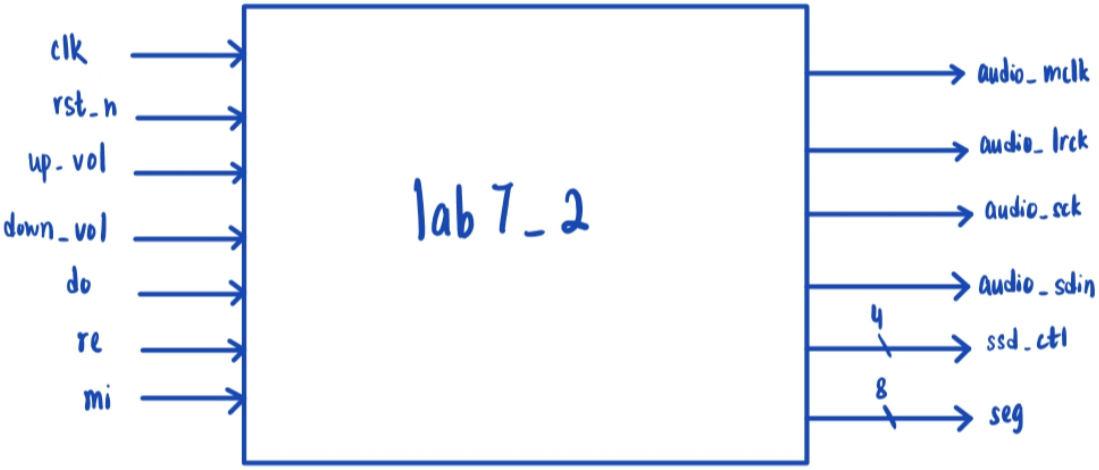
input clk, rst_n;
input [15:0] audio_in_left, audio_in_right;
output audio_mclk, audio_lrck, audio_sck, audio_sdin;



The input clk is the 100MHz clock.
The rst_n is a reset.
The input each 16 bits of audio_in _left and audio_in_right are the input from the buzzer control. But since the lab7_1 only need to implement the parallel to serial converter, so I didn't include the buzzer control in this section.
The output is designed as requested, which is the audio_mclk, audio_lrck, audio_sck and audio_sdin.
The audio_mclk(Master Clock) synchronizes the audio data transmission, which is 25MHz.
The LRCK (Left-Right Clock, or Word Select (WS) Clock, or Sample Rate (Fs) Clock) controls the sequence (left or right) of the serial stereo output, which is 25MHz/128, nearly 192kHz.
The Serial Clock (SCK) controls the shifting of data into the input data buffers, which is 25MHz/4, 6.25MHz.

Lab7_2:

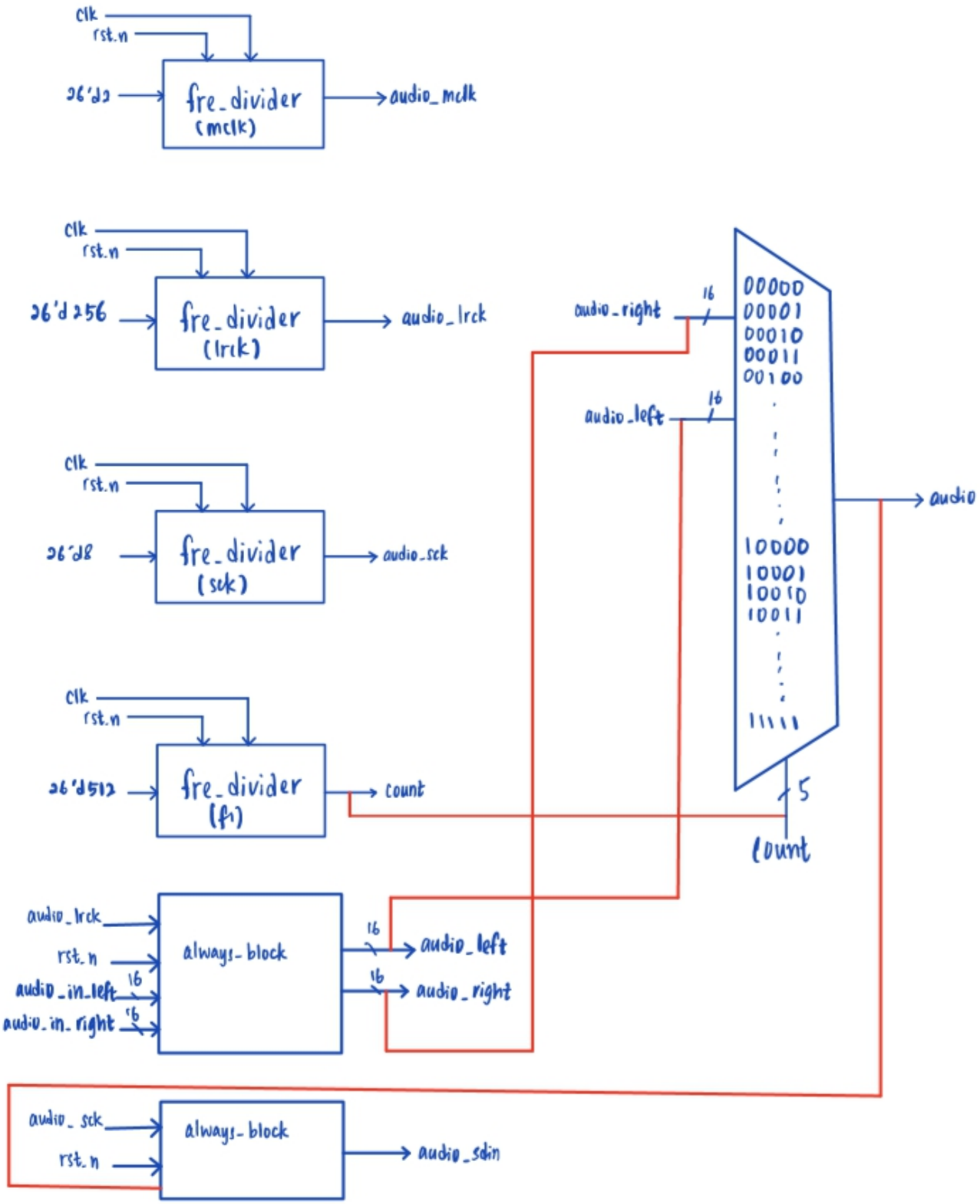
input clk;
input rst_n;
input up_vol;
input down_vol;
input do, re, mi;
output [3:0] ssd_ctl;
output [7:0] seg;
output audio_mclk, audio_lrck, audio_sck, audio_sdin;



The input clk is the 100MHz.
The input rst_n is a reset.
The input up_vol is the button control for increase the volume of the tune.
The input down_vol is the button control for decrease the volume of the tune.
The input do, re, mi is the button for control which tune(do, re, mi), to be played.
The 4 bits ssd_ctl output is the seven segment decoder display, while output of 8 bits of seg decide which segment of the ssd to be displayed.
The output audio_mclk, audio_lrck, audio_sck, audio_sdin designed as requested for the speaker to function.
As stated above, the audio_mclk(Master Clock) synchronizes the audio data transmission, which is 25MHz.
The LRCK (Left-Right Clock, or Word Select (WS) Clock, or Sample Rate (Fs) Clock) controls the sequence (left or right) of the serial stereo output, which is 25MHz/128, nearly 192kHz.
The Serial Clock (SCK) controls the shifting of data into the input data buffers, which is 25MHz/4, 6.25MHz.

Design Implementation

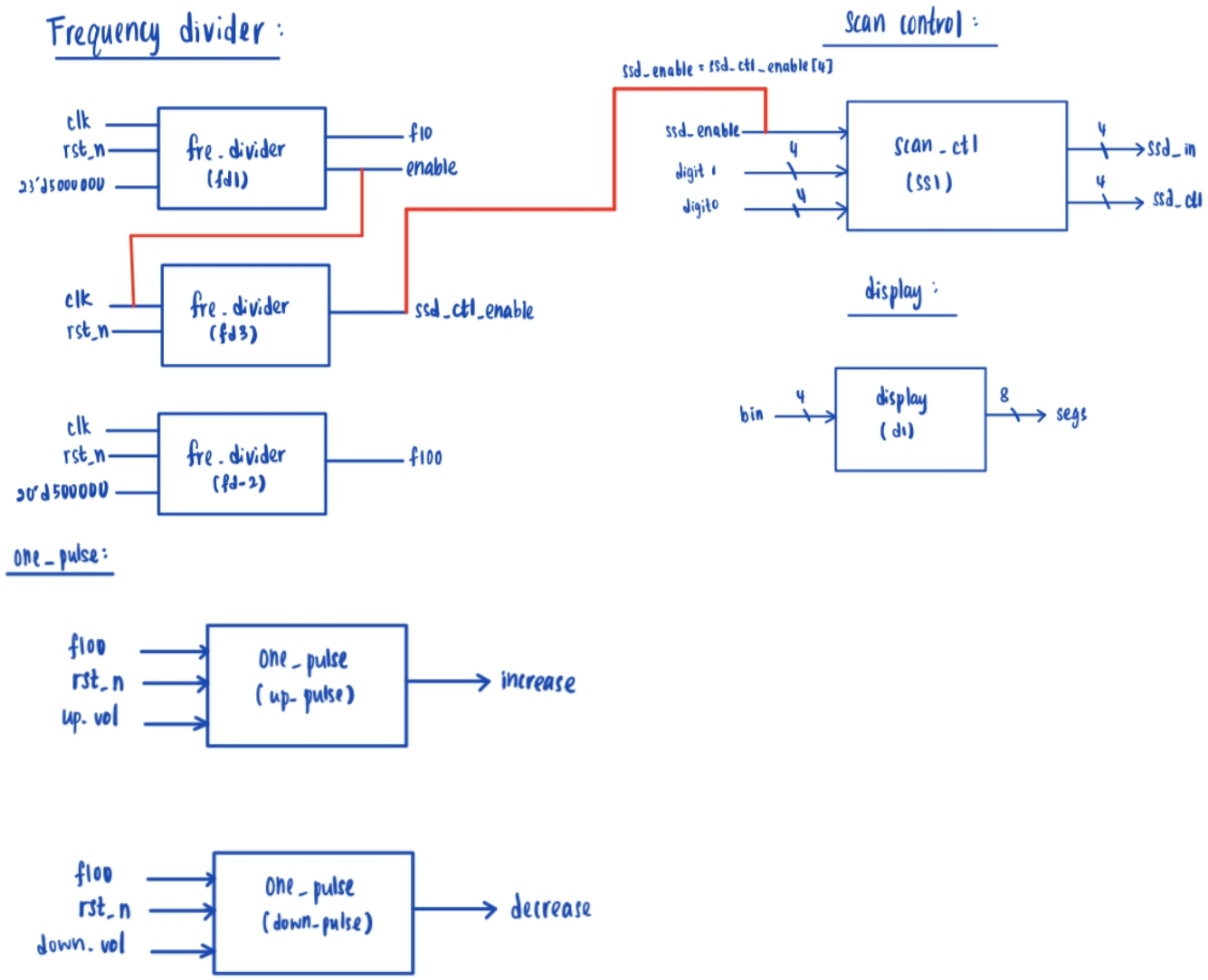
Lab7_1:



There are 4 frequency divider used in lab7_1, as to generate the four different output, which are audio_mclk, audio_lrck, audio_sck, audio_sdin. The audio_mclk is generate by the 26'd2 counter, audio_lrck is generate by a 26'd256 counter, the audio_sck, is generate by a 26'd8 counter while the audio_sdin is generate by the parallel to serial converter. The forth

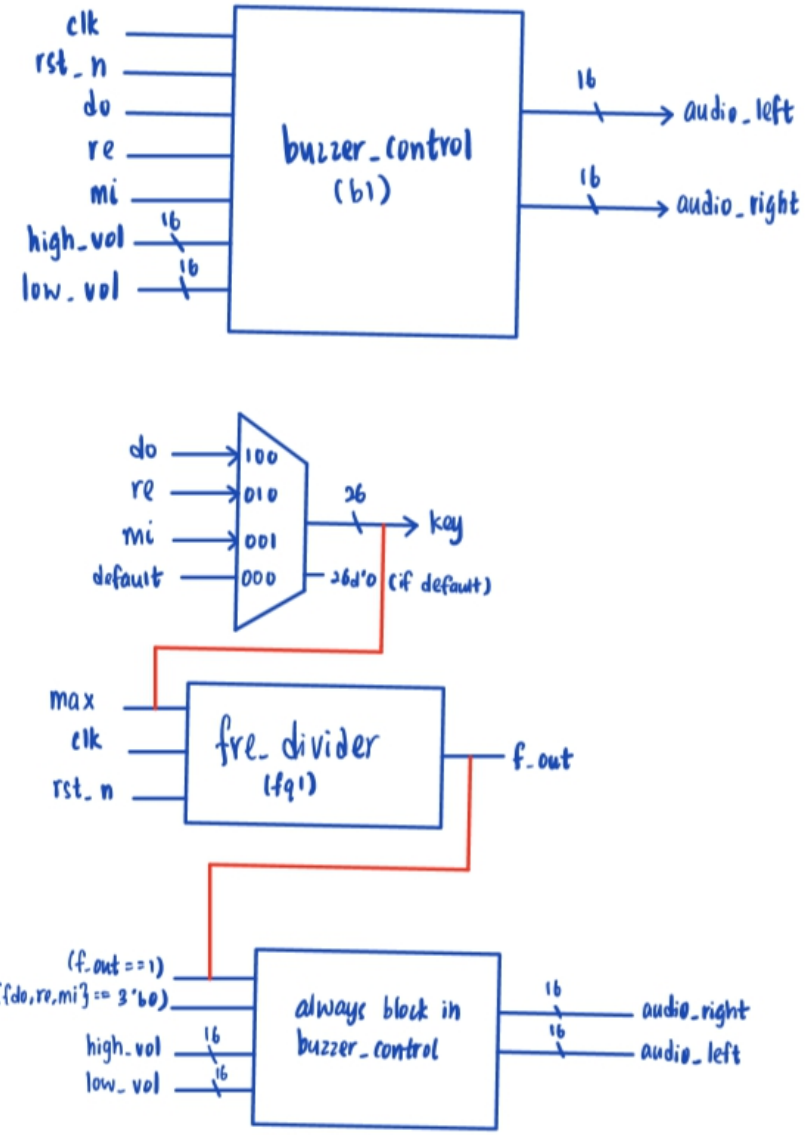
frequency divider uses 26'd512 counter to generate a count(output) to act as the select for the mux of the parallel to serial converter. The mux will generate the serial output called audio, while the audio_left(audio_in_left) and the audio_right(audio_in_right) will act as the input to the mux. The audio output is then pass to the audio_sdin.

Lab7_2:



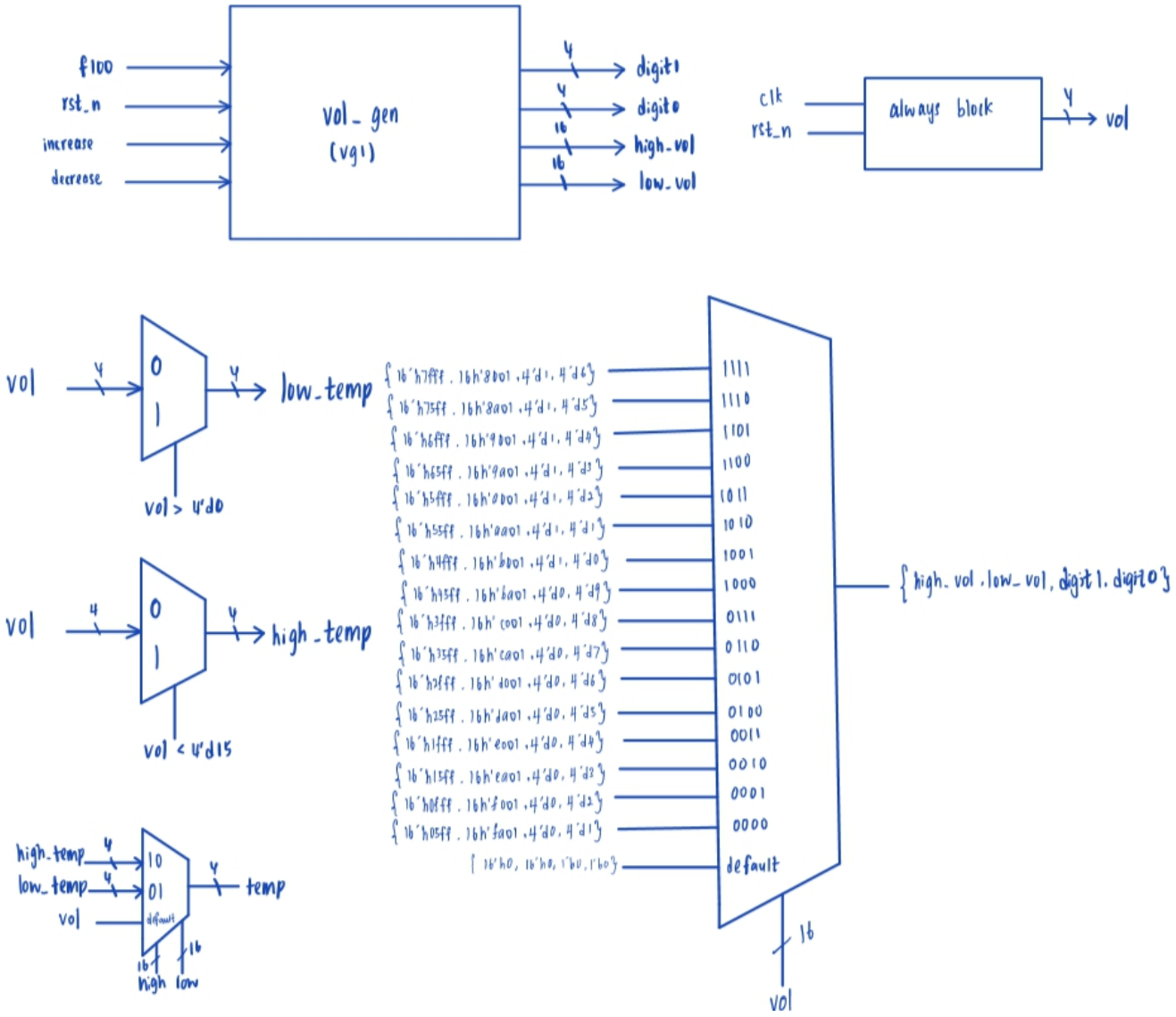
The frequency divider fd1 generate a 10Hz clock, which is divided by the 23'd5000000 counter, and an enable signal which then passed to the fd3 as clk to generate the control for scan control, which called ssd_ctl_enable. The frequency divider fd2, is use to generate a 100Hz clock, by using a 26'd500000 counter. The output of 100Hz clock is called f100, which is used to control the volume increase, volume decrease and generate the different frequency of tune of do, re, mi.
The f100 clock will connect to the 2 one_pulse for changing the volume by increasing and decreasing according to the times that button is being pressed.
The ssd_ctl is used to determine which of the seven segment decoder to be used, and the ssd_in will pass to the display. The display will decide which segments are going to be used to display the digit required according to the input bin(ssd_in).

Buzzer control :



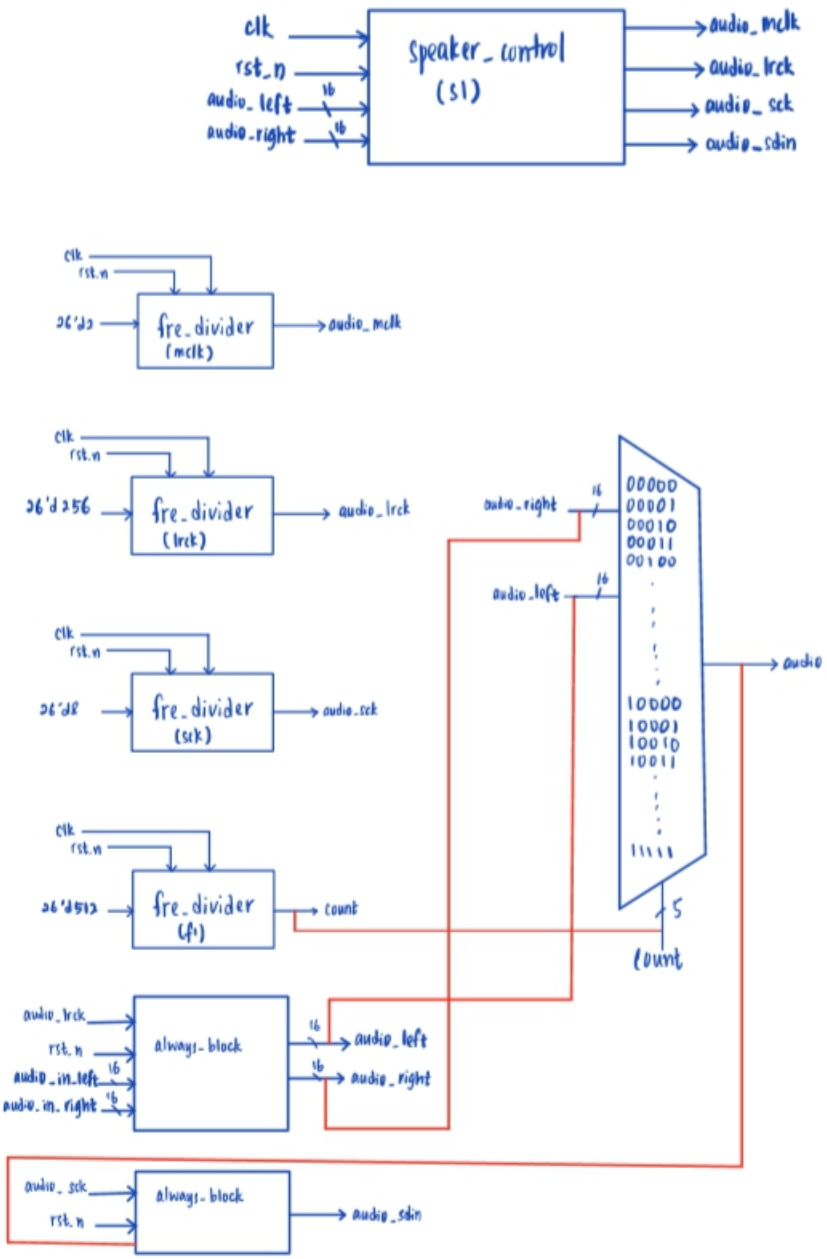
The buzzer control is used to generate the tone, do, re, mi, and then generate the output of audio_left and audio_right to be used in the speaker control. In which both audio_left and audio_right are 16 bits.

volume generator :



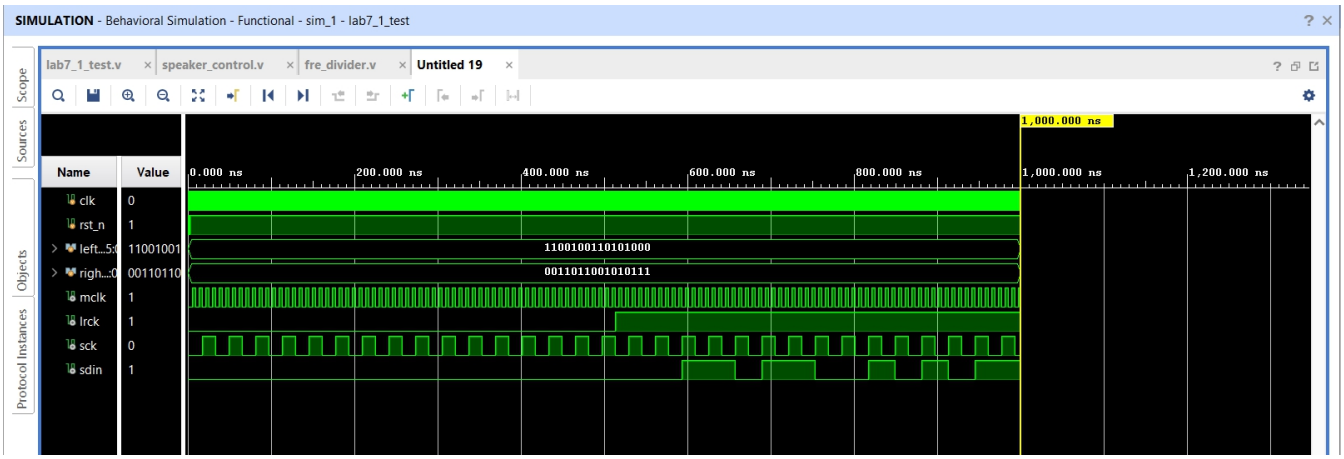
The vol_gen is used to generate the different tune(or frequency) of do, re, mi, when the volume is change either increasing or decreasing. The volume change is then generate a different frequency of tune of do, re, mi.

Speaker control:



The speaker was done in lab7_1.
As stated above, the output from the buzzer control will send to the speaker control as inputs, which is the 16 bits of audio_in_left and audio_in_right.
The audio_mclk is generate by the 26'd2 counter, audio_lrck is generate by a 26'd256 counter, the audio_sck, is generate by a 26'd8 counter while the audio_sdin is generate by the parallel to serial converter. The forth frequency divider uses 26'd512 counter to generate a count(output) to act as the select for the mux of the parallel to serial converter. The mux will generate the serial output called audio, while the audio_left(audio_in_left) and the audio_right(audio_in_right) will act as the input to the mux. The audio output is then pass to the audio_sdin.

Discussion



The wave form shown above is the result of the simulation for lab7_1.
The design is that audio_left as left = 16'b1100_1001_1010_1000; and audio_right as right = 16'b0011_0110_0101_0111;

From this lab 7, I had learned how to generate different tune of do, re, mi by different frequency. The other things that I had learned is that how to generate different frequency divider as to use for the speaker.
Although it is hard, but I still think that I did learn something from this lab.

Conclusion

I had learned how to use different frequency to generate different tune of do, re, mi. I think that I did learned the timing control for the speaker.

