# Logic Design Lab 7

## Experiment 1

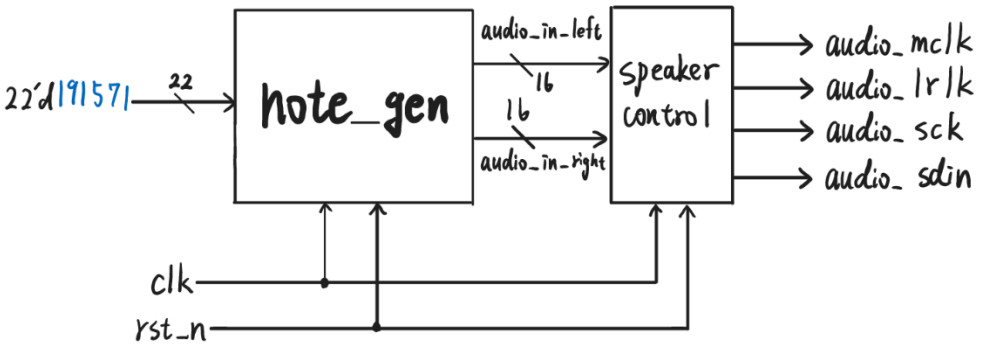
***Please design an audio-data parallel-to-serial module to generate the speaker control signal with 100MHz system clock, 25 MHz master clock, (25/128) MHz Left-Right clock(Fs), and 6.25 MHz (32Fs) sampling clock.***

### Design Specification：

Input: clk, rst_n.

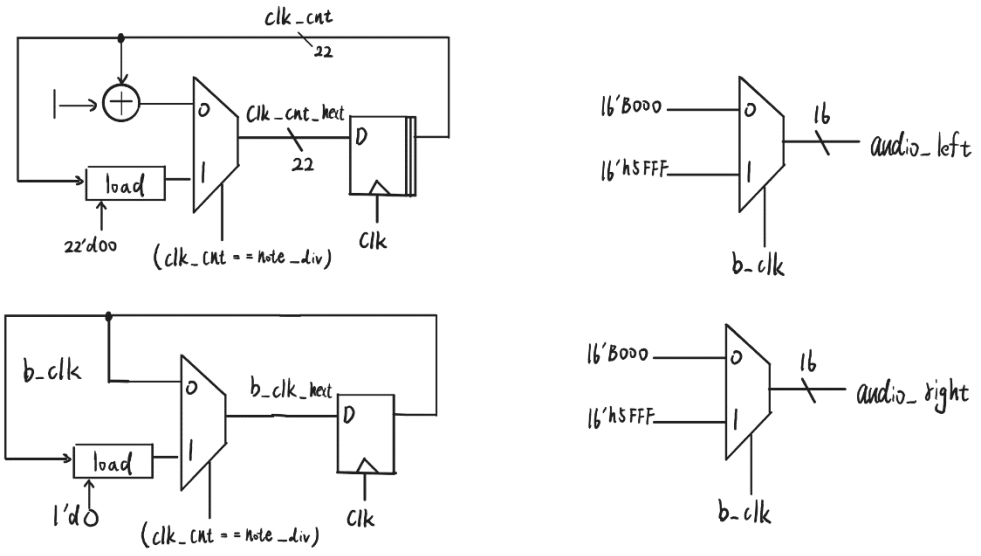
Output: arduino_mclk, arduino_lrlk, arduino_sck, arduino_sdin.

Block diagram:

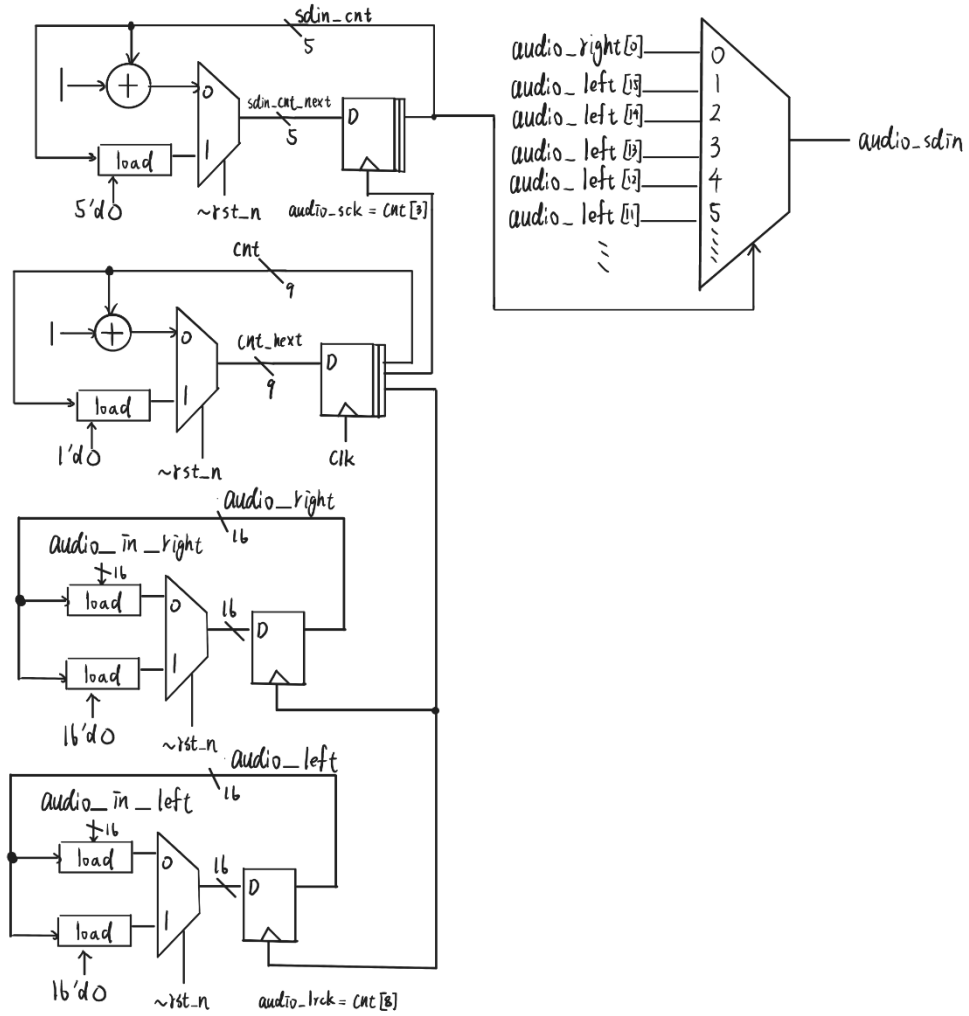



### Design Implementation：

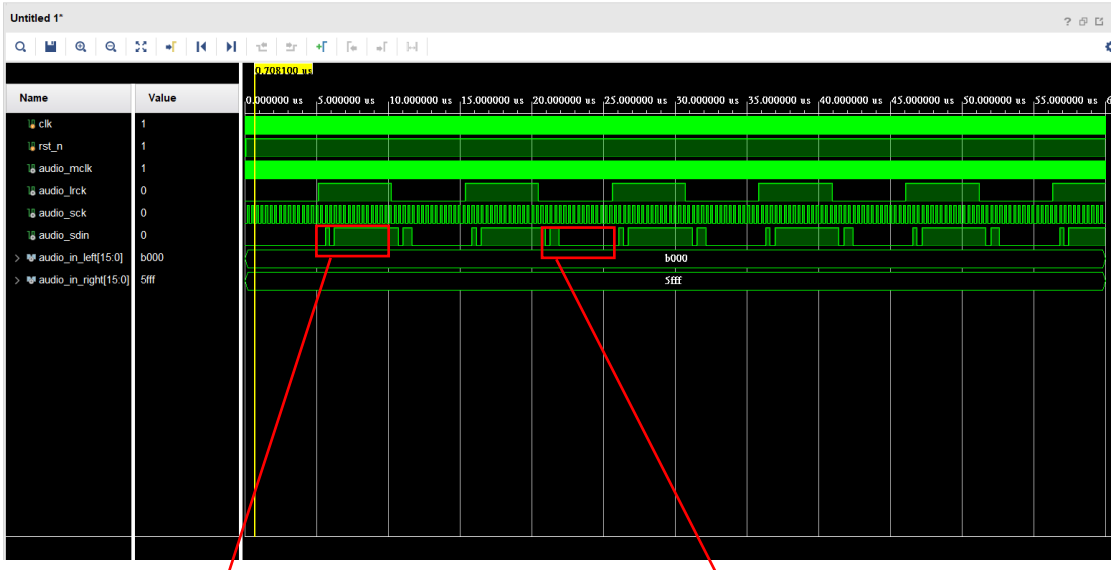
1. Note generator：



2. Speaker control：




3. Simulation wave form




16b'1011111111111111 = 16h'5fff

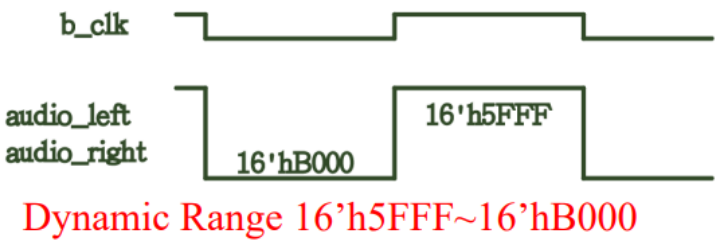
16b'1011000000000000 = 16h'b000

## Discussion：

1. Note generator：
   (1) We use a counter with some limit to generate the buzzer frequency (buzzer clock) we want. For example, if we want to get the buzzer clock of Do (frequency=261 HZ), we calculate the value $191571 = {}^{100M}/_{2 * 261}$.

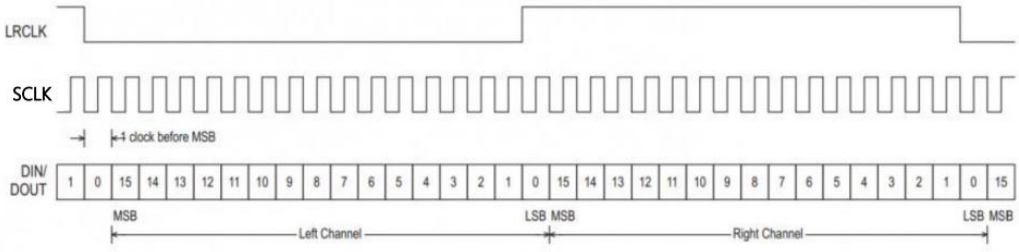
   (2) The reason why we can use the formula is that we use the original clock from the Basys3 board is 100MHZ, and an ideal clock cycle include two period for high and low.

   (3) After we gain the correct buzzer clock, we can decide the volume of the wave form by design the amplitude of the wave in the range of [16h'5fff, 16h'b000].




(from the handout)

2. Speaker control：
   (1) First, we get master clock, left-right clock, and serial clock from the counter generated by the original clock. And then we can decide the input and output relation from different clocks according to the block diagram.
   (2) Then, we can use a mux to assign the correct value to serial data input according to the block diagram and the following plot from the handout.




3. From the simulation, I write a testbench where the amplitude of the left channel is 16h'b000 while the amplitude of the right channel is 16h'5fff. The result from the process of parallel to serial seems correct.

## Conclusion：

This experiment is the fundamental model of this lab. If we can turn the parallel information to the serial data with 3 clocks correctly, the latter 3 experiment can be done with more ease. Besides, through this experiment, we can have better insight of the communication and transmission of data. Hope it will be helpful to advance courses.

# Experiment 2

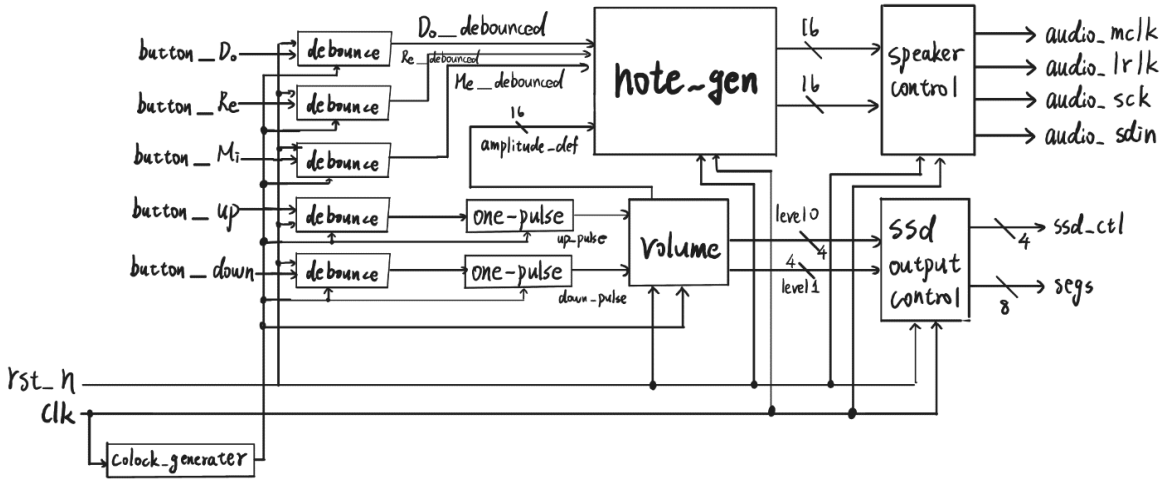
***Speaker control.***

## Design Specification：

Input: clk, rst_n, button_Do, button_Re, button_Me, button_up, button_down.

Output: arduino_mclk, arduino_lrlk, arduino_sck, arduino_sdin, ssd_ctrl [3:0], display [7:0].

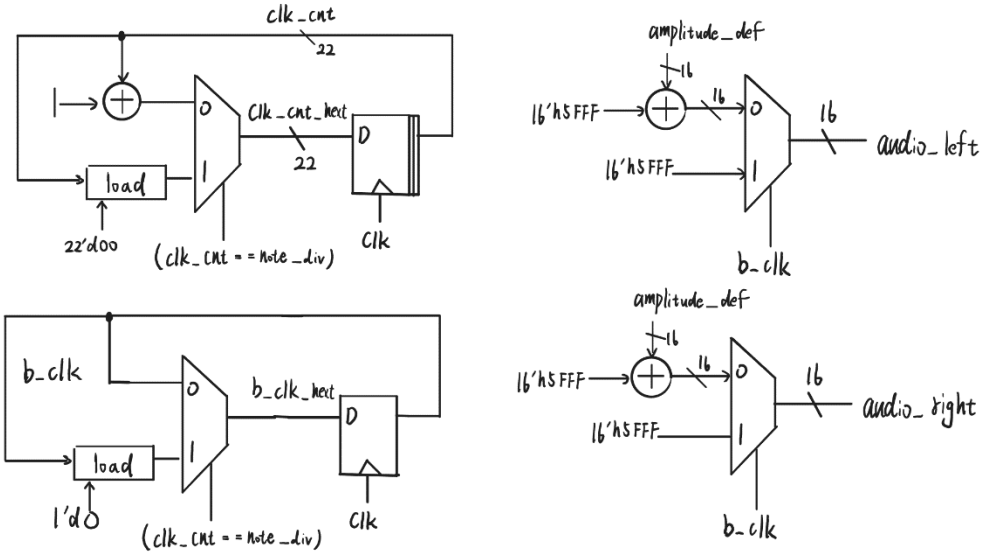
Block diagram:



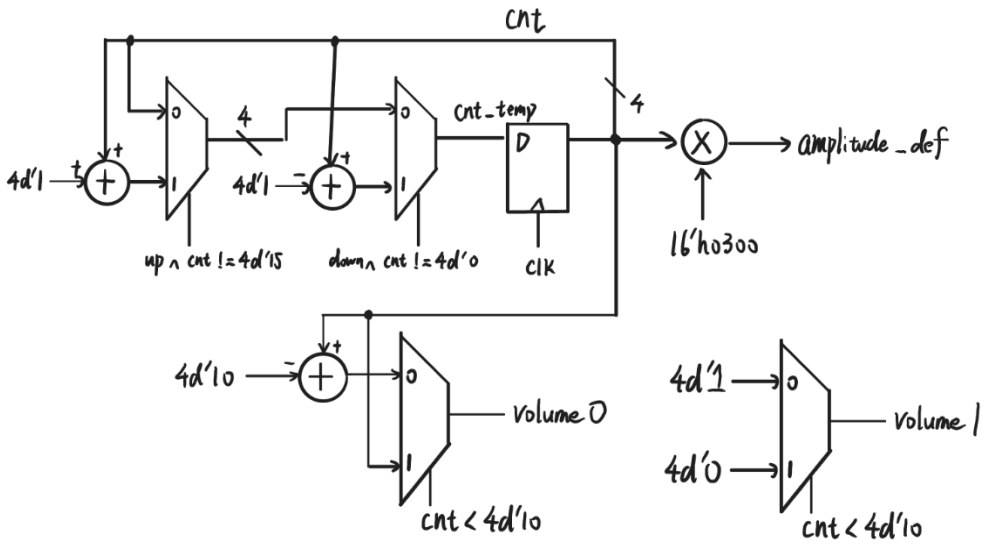

## Design Implementation：

1. Debounce and one pulse：
   It has been implemented in the previous labs. Skip.

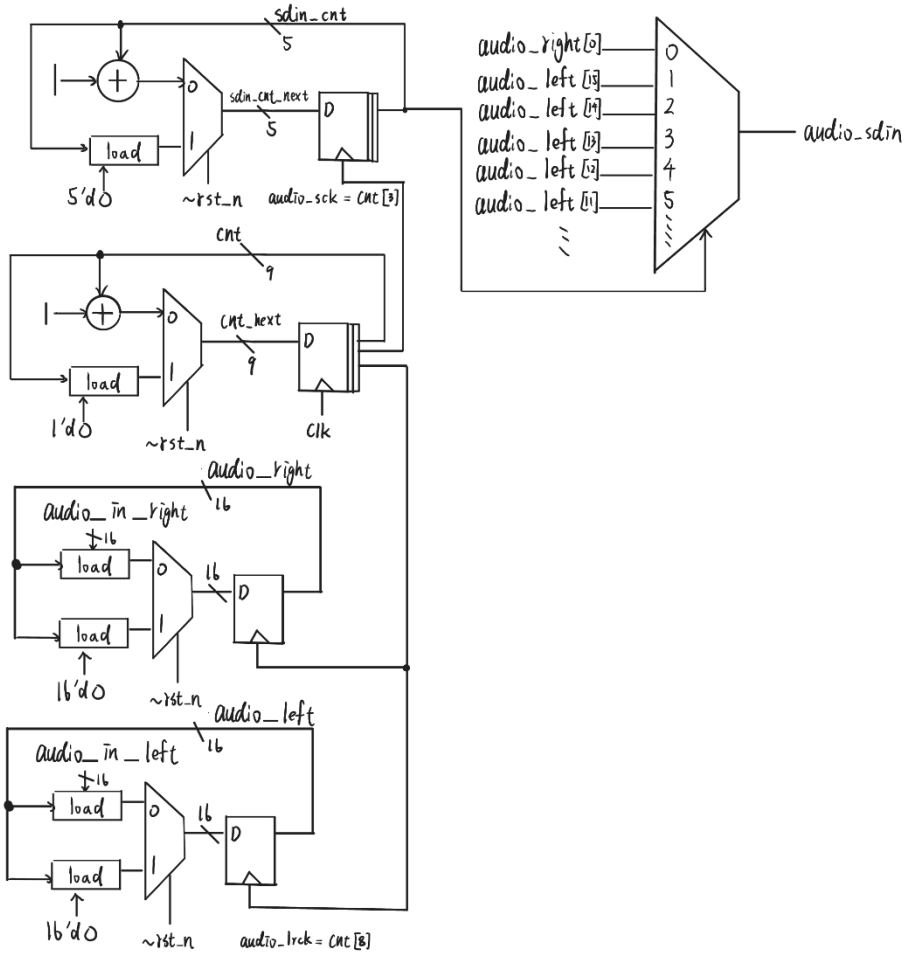
2. Note generator：



3. Volume：




4. Speaker control：



## Discussion：

1. I change the design of the note generator and add a block volume to make it turn the volume up or down.
2. Because the dynamic volume range is [16h'5fff, 16h'b000], we can assign the amplitude difference $16\text{h'}0555 = {16\text{h'}\text{b}000 - 16\text{h'}5\text{fff}}/{15}$ since we have 16 different volumes.
3. However, perhaps, because of the maximum volume is too large to what the headphone can withstand, which leads to the volume changing becomes less and less, so I set the amplitude difference $16\text{h'}0300$ to deal with the problem

## Conclusion：

This experiment took me the largest time in this lab, because first, we need to check the code written in the lab7_1 is well-done through the result from the board practically. besides, we need to not only change the volume of the speaker but also check there aren't any bugs in the other blocks or function calls. However, after finishing the experiment, I had basic knowledge of the implementation of the speaker, which allows me to get faster un the latter experiments.

# Experiment 3

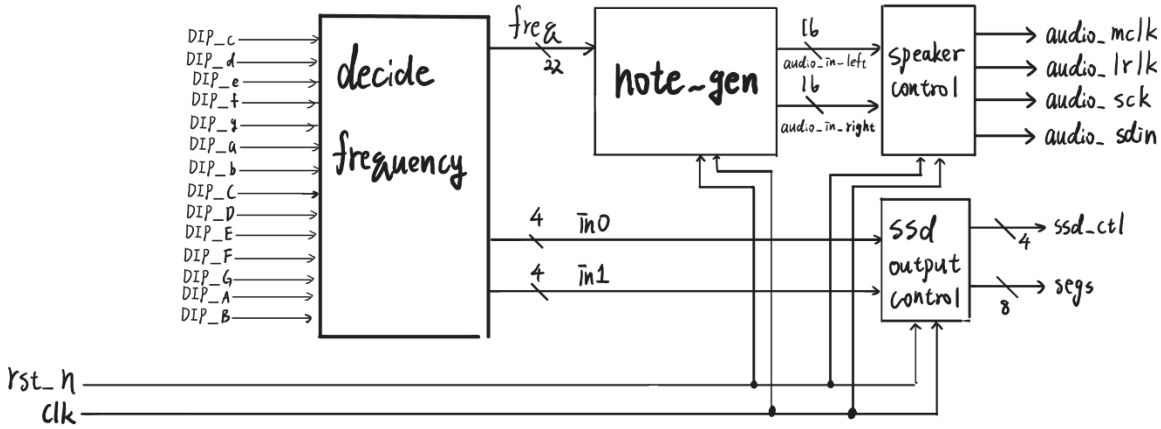
***Electronic Organ.***

## Design Specification：

Input: clk, rst_n, DIP_c, DIP_d, DIP_e, DIP_f, DIP_g, DIP_a, DIP_b, DIP_C, DIP_D, DIP_E, DIP_F, DIP_G, DIP_A, DIP_B.

Output: arduino_mclk, arduino_lrlk, arduino_sck, arduino_sdin, ssd_ctrl [3:0], display [7:0].

Block diagram:




## Design Implementation：

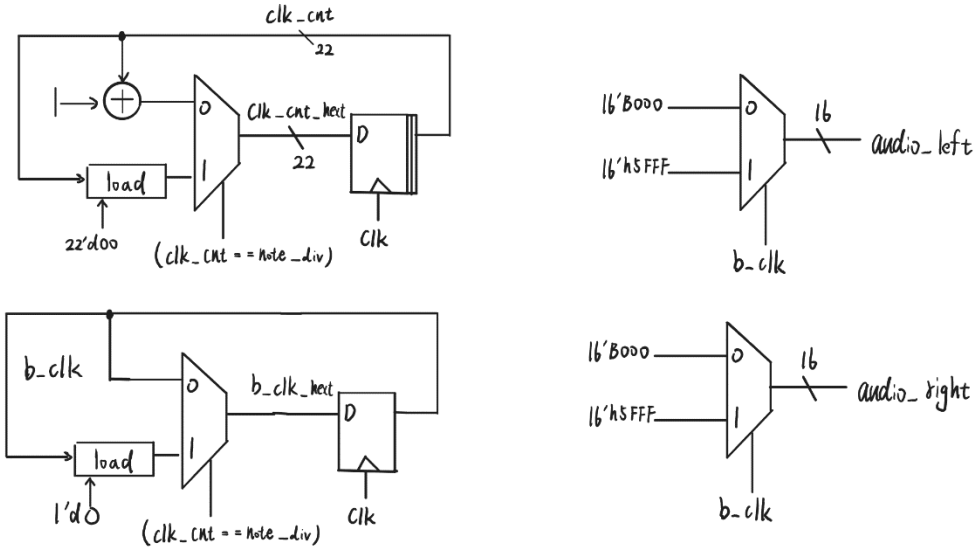
1. Decide frequency：

   I implement it with if/else condition with default condition. (Refer to the screen shot.)

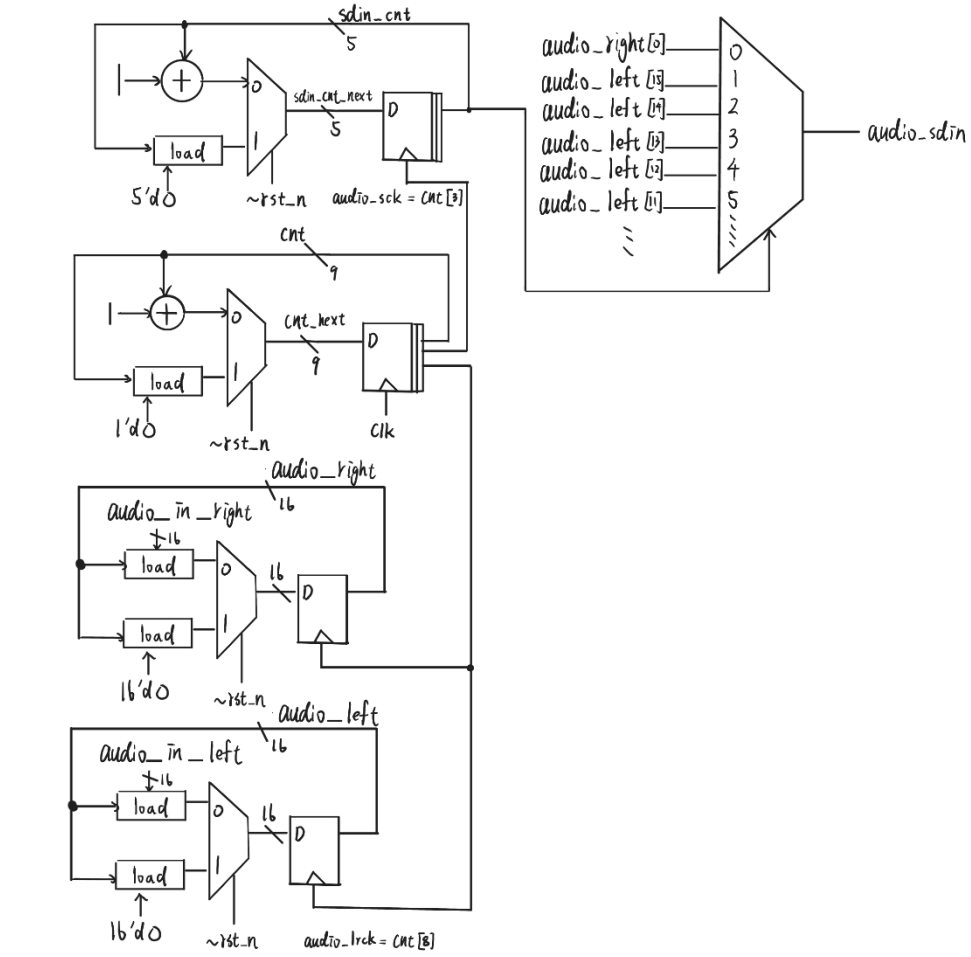
```
31  always@(DIP_c, DIP_d, DIP_e, DIP_f, DIP_g,DIP_a, DIP_b,
32          DIP_C, DIP_D, DIP_E, DIP_F, DIP_G,DIP_A, DIP_B)begin
33    if (DIP_c & ~DIP_d & ~DIP_e & ~DIP_f & ~DIP_g & ~DIP_a & ~DIP_b &
34          ~DIP_C &~DIP_D &~DIP_E & ~DIP_F  &~DIP_G & ~DIP_A & ~DIP_B)
35      begin
36      freq= `freq_mid_Do;
37      in0 = 4'd12;
38      in1 = 4'd4;
39      end
40    else if (~DIP_c & DIP_d & ~DIP_e & ~DIP_f & ~DIP_g & ~DIP_a & ~DIP_b &
41          ~DIP_C &~DIP_D &~DIP_E & ~DIP_F  &~DIP_G & ~DIP_A & ~DIP_B)
42      begin
43      freq= `freq_mid_Re;
44      in0 = 4'd13;
45      in1 = 4'd4;
46      end
47    else if (~DIP_c & ~DIP_d & DIP_e & ~DIP_f & ~DIP_g & ~DIP_a & ~DIP_b &
48          ~DIP_C &~DIP_D &~DIP_E & ~DIP_F  &~DIP_G & ~DIP_A & ~DIP_B)
49      begin
50      freq= `freq_mid_Me;
51      in0 = 4'd14;
52      in1 = 4'd4;
53      end
```

```
132    else
133      begin
134      freq= 4'd0;
135      in0 = 4'd0;
136      in1 = 4'd0;
137      end
138  end
```

2. Note generator：




3. Speaker control：



## Discussion：

1. We use a mux (in fact, a series of if else conditions) to decide the relation between DIP switch and the frequency we want.
2. The other parts of the speaker is the same as the lab7_1.

# Experiment 4

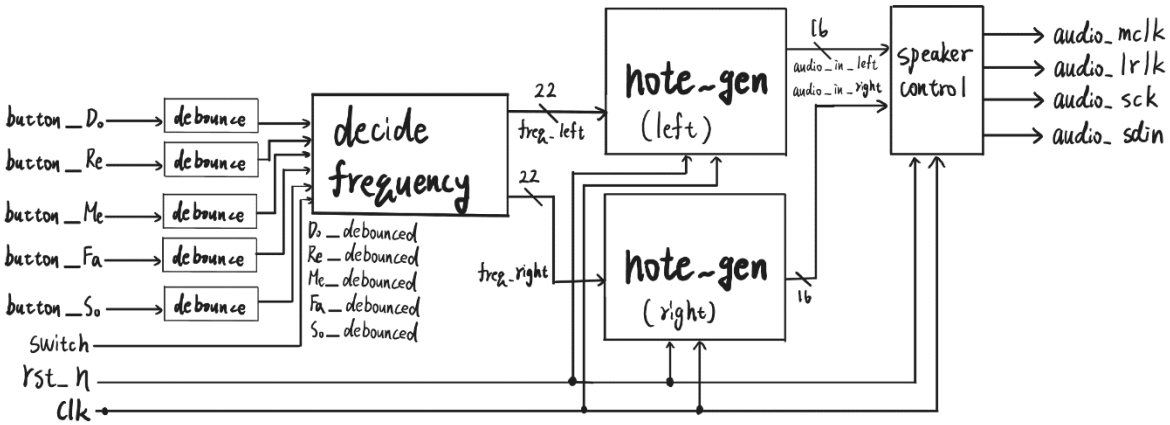
***Playback double tones by separate left and right channels. If you turn one DIP switch off, the electronic organ playback single tone when you press push button. If you turn DIP switch on, left (right) channels play Do(Mi), Re(Fa), Mi(So), Fa(La), So(Si) when you press the five push buttons, respectively.***

## Design Specification：

Input: clk, rst_n.

Output: arduino_mclk, arduino_lrlk, arduino_sck, arduino_sdin, ssd_ctrl [3:0], display [7:0].

Block diagram:

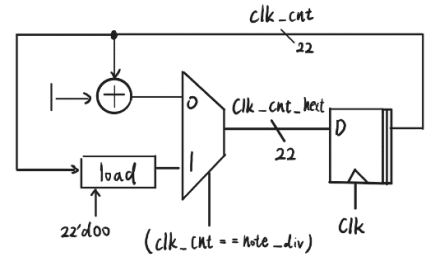


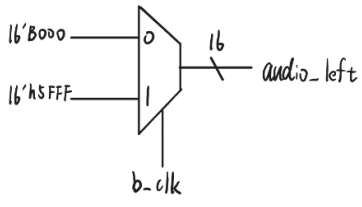

## Design Implementation：

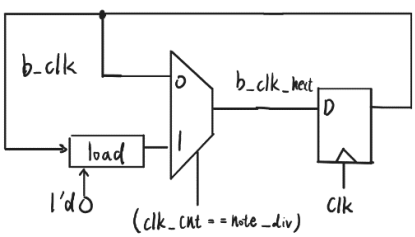
1. Decide frequency：

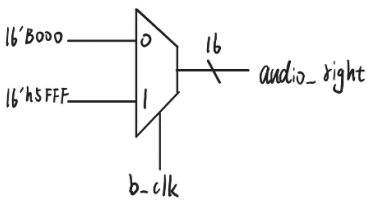
   The implementation is similar to the lab7_3. Skip.

2. Note generator：

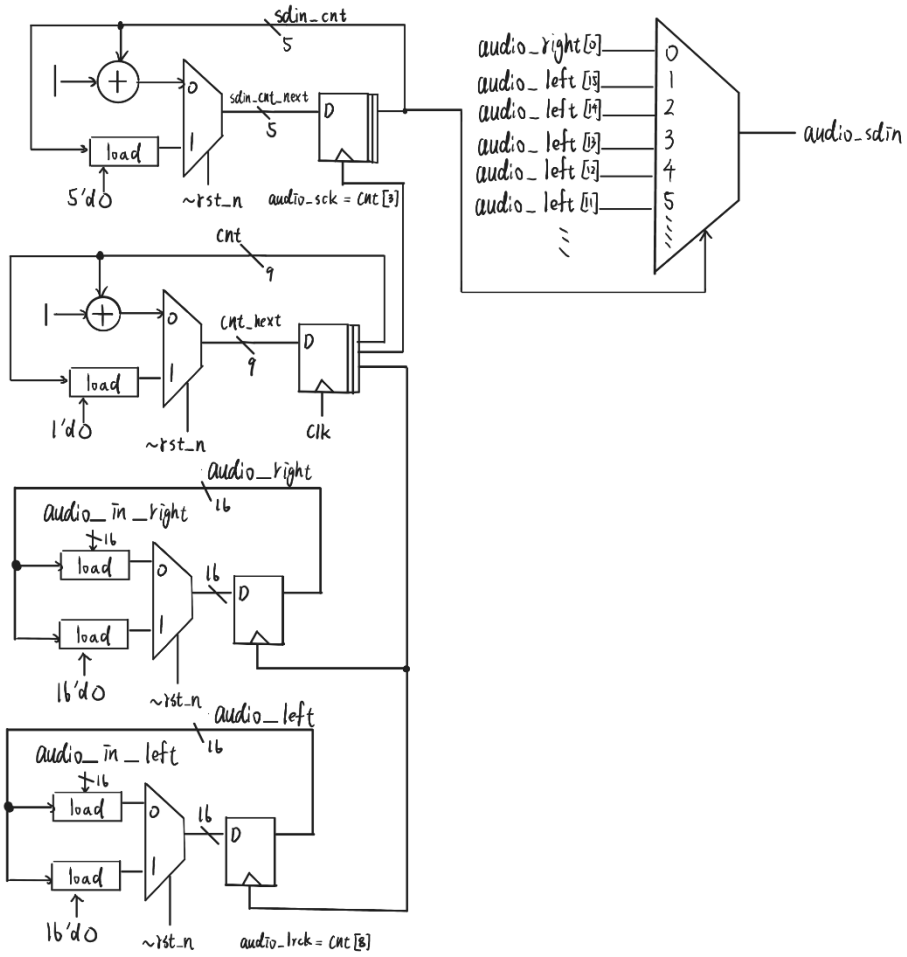


3. Speaker control




## Discussion：

1. The only difference between this lab and the previous labs is that we need to allow our speaker to have right/left channels.
2. It can be implemented easily by copying the note generator again. By doing so, we can have two not generators to control the frequency of right and left channels.

## Conclusion：

This experiment is based on the previous lab, by the method discussed in the discussion, we can get the speaker with different right/ left channels we want.

## Reference：

1. The handout and the assignment of logic design
To review some basic concept of the combination circuit and the problem bulls and cows.

2. The handout of logic design lab

To program our FPGA board by following the method on the handout of logic design lab.