CS 2104 02 Hardware Design and Labs 2018
Final Project Report

ID:106062119 Name:王元廷

1. 設計概念

As illustrated in proposal, the goal of my final project is to implement a piano with metronome functions. In general, I use keyboard and audio amplifier to implement the piano with the LEDs and 7-segment displays begin the part of metronome.

- Piano

  - The keys highlighted in red and black represent an octave.
  - Change the pitch of the octave using the arrow keys.
    (As in the picture, the up-down arrows modify the upper octave and
    the left-right arrows modify the lower octave)
  - The audio amplifier plays the corresponding sound pressed by the keyboard

    > I was planning to make it stereo by using some techniques; unfortunately, it failed so there is only mono sound track.

- Metronome

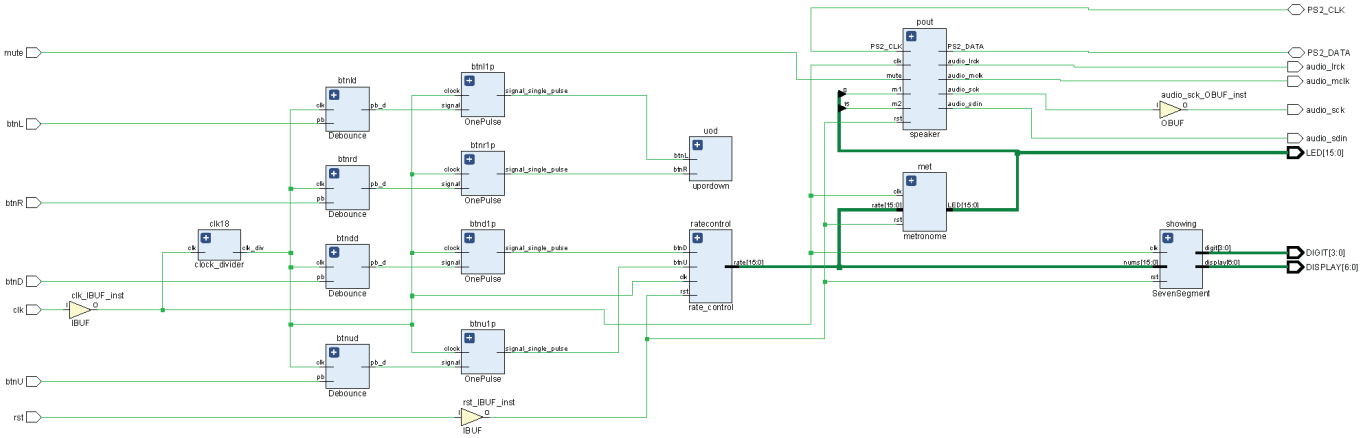
  - LEDs serve as a metronome which shift from left to right or from right to left as a cycle based on the BPM (beats per minute) value.
  - The BPM value will be displayed on the three rightmost 7-segment displays and can be modified using BTNU and BTND.

    > The leftmost displayed was initially reserved for the octave representation, but an awkward timing loop occurs in implementation so I removed it unfortunately.

  - A sound of 880Hz will be displayed when LED is at both sides (meaning that a cycle starts/ends).
  - The sound of metronome could be mute using the leftmost switch.

2. 架構細節及方塊圖

The block diagram is displayed below.



piano

This is the top module of the project, so it is not displayed in the above diagram. The required input signals are **clk**, **rst**, **btnU**, **btnD**, (**btnL**, **btnR**); output signals are **LED**s for LED handling, **DISPLAY** for 7-segment display, and the four audio output signals; inout signals **PS2_DATA** and **PS2_CLK** for keyboard design.

> **btnL** and **btnR** are initially designed to switch whether to display the above or lower octave.

Onepulse/Debounce

Like previous labs, all button signals are implemented in one-pulse form.

rate_control

```
//module rate_control(btnU, btnD, rate, clk, rst);

    always @* begin
        next_rate = rate;
        if(btnU==1'b1 && rate!=16'h0218) begin
            next_rate = rate + 1;
            if(rate[3:0]==4'h9) next_rate = rate + 7;
            if(rate[7:0]==8'h99) next_rate = rate + 103;
        end
        if(btnD==1'b1 && rate!=16'h0040) begin
            next_rate = rate - 1;
            if(rate[3:0]==4'h0) next_rate = rate - 7;
            if(rate[7:0]==12'h00) next_rate = rate - 103;
        end
    end
```

For the convenience of showing displays, *rate* is assigned as **wire** [**15**:**0**], which corresponds [**15**:**12**], [**11**:**8**], [**7**:**4**] and [**3**:**0**] to the digit with thousands, digit with hundreds, digit with tens and digit with ones respectively.

Therefore, it requires a bit more actions dealing with the arithmetic stuff. Besides, highest and lowest bounds of the rate are set to **218** and **40** to make it the same as the real metronome.

`metronome`

```
// module metronome(rst, clk, rate, LED);
    always @* begin
        next_dir = dir;
        if(cnt>=(400_000_000/real_rate)-1) begin
            next_cnt = 34'b0;
            if(dir==1'b0) begin
                next_LED = LED << 1;
                if(LED==16'h0000) begin
                    next_LED = 16'h4000;
                    next_dir = 1'b1;
                end else if(LED==16'h4000) begin
                    next_dir = 1'b1;
                end
            end else begin
                next_LED = LED >> 1;
                if(LED==16'h0000) begin
                    next_LED = 16'h0001;
                    next_dir = 1'b0;
                end else if(LED==16'h0002) begin
                    next_dir = 1'b0;
                end
            end
        end else begin
            next_cnt = cnt + 1;
            next_LED = LED;
        end
    end
```

Since the frequency of the FPGA is **100**MHz, the upper bound of *cnt* is set to be ((100,000,000 * 60) / 15) / real_rate = **400,000,000**/*real_rate*.
Where *real_rate* is assigned as follows.

```
  wire [8:0] real_rate = (rate[3:0] + (rate[7:4]*10) + (rate[11:8]*100));
```

So whenever the metronome counts to that upper bound, the LED will shift either right or left by one based on *dir* signal. When LED reaches the leftmost/rightmost edges, *dir* changes to make the shifting inverse.

> There is an awkward bug that I could not figure it out even before the time of demo.
> Sometimes the LEDs will become **16'h0000** (all off), when the rate changes too fast.
> There is no specific *dir* or pattern that would make it disappear in my observation. That is, the bug appears `randomly` when the rate changes too fast.
> Since I cannot figure out the reason, I have no idea about how to handle this bug. So at the end I

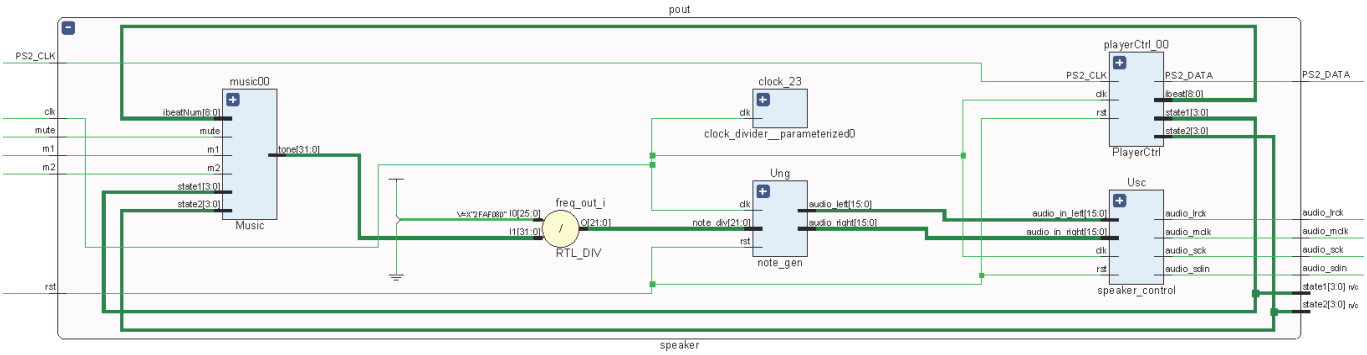
> simply make the LEDs start from **16'h8000** or **16'h0001** again when they all disappear, which does not exactly fix it.

`SevenSegment`

This module is almost the same as the one in lab06 sample code. The only change is that when the thousands and hundreds digits of *rate* == **0**, it will display nothing.

`speaker`

This module handles all the piano part as well as making the speaker plays the metronome rate. It is quite complex, so the detailed diagram is shown below.



- `PlayerCtrl` outputs *ibeat*, based on *PS2_DATA* and *PS2_CLK*, which means that keyboard directly controls the *ibeat* values.
- `PlayerCtrl` additionally outputs *state1* and *state2*, indicating which octave the two scales are in.
- `Music` gets the *ibeat* as well as *state1*/*state2* values to know that which *tone* value is to output.
- `note_gen` and `speaker_control` are the same as previous labs, handling signals from `Music` and make the audio amplifier play corresponding sound.

> The reason why *state1* and *state2* values are assigned as output in *speaker* is that I planned to show them on the 7-segment display, but in vain.

3. 實作完成度/難易度說明/分工

I have completed 80% of my proposal. The only part missing is the *btnL* and *btnR* part. The part aims to select which octave to display on the 7-segment displays, the upper one or the lower one.
Besides, I had planned to make some enhancement. First, which is the easiest part, is to enable/disable the sound of the metronome. I placed it on the leftmost switch. Second, I had been searching for how to mix several sound in a mono sound track. Standard headphones are stereo, but we can hear all kinds of sound, which is obviously larger than two sounds. However, after some testing and trying, the branch design plays only trash sound. Having not enough time, I finally aborted the feature.
To conclude, the difficulty of the basic features is moderate. Some testing features are much more difficult, while I wasted a lot of time but did not success in doing the implementation.
As for the job allocation part, I have made all parts of my project.

4. 測試完整度

Since the entire project is not quite complicated, I have shown all parts of my project during the demo part. In addition, we are required to turn the related devices back right after the demo, so I had no chance to improve

my project after the demo.

5. 困難與解決方法

> The difficulties and solutions I faced have been illustated above in this type of section syntax. Thanks.