

## Lab 8: Electric Piano

### Submission Due Dates:

Demo: 2021/12/21 17:20  
Source Code: 2021/12/21 18:30  
Report: 2021/12/26 23:59

### Objective

- Getting familiar with the audio peripheral and Pmod Connector.

### Description

After watching an anime called "Your Lie in May," your best friend has a dream to become a great pianist. However, he or she can't afford to buy a piano to practice. As an NTHU CS student, you want to make your best friend's dream come true. You decide to use the knowledge learned in this class to design an electric piano with a computer keyboard as an instrumental keyboard.

- The piano has two modes:
  - PLAY: In this mode, the user can play the piano using the keyboard.  
DEMONSTRATE: In this mode, the piano can demonstrate a song that has been written in it. So, the player can get to know how the song sounds like.

- In the PLAY mode, the piano should make no sound unless you press a key. The following table shows the key-note mapping.

| Key  | a         | s         | d         | f         | g          | h         | j         |
|------|-----------|-----------|-----------|-----------|------------|-----------|-----------|
| Note | C (or Do) | D (or Re) | E (or Mi) | F (or Fa) | G (or Sol) | A (or La) | B (or Ti) |

If you think these notes are not enough to play a good song, feel free to add more on other keys.

- In the PLAY mode, the piano should play a note when you press a corresponding key and stop when you release it. Only one note will be played at a time. You may define the behavior if multiple keys are pressed at once.
- In the DEMONSTRATE mode, the piano will **repeatedly** play a demonstration song (starting from the beginning again when reaching the end) and will NOT react to any key pressed on the keyboard. You can pick the song from the sample sheets below
- The piano should support two tracks. In the PLAY mode, two tracks play the same note. In the DEMONSTRATE mode, one track plays the melody part, and the other plays the accompaniment part.
- The piano should support 5 distinguishable levels of volume, controlled by **Volume Up** (BtnU) and **Volume Down** (BtnD). The default level of volume is 3. At the lowest level, the sound should still be able to heard. Pressing **Volume Up** at level 5 or pressing **Volume Down** at level 1 takes no effect.
- The piano should support 3 levels of octaves, controlled by **Higher Octave** (BtnR) and **Lower Octave** (BtnL). The default level of the octave is 2. Pressing **Higher Octave** at level 3 or pressing **Lower Octave** at level 1 takes no effect.  
Raising a note an octave higher means to double the note's frequency, and vise versa.  
(ex. A4 with frequency 440 Hz -> A5 with frequency 880 Hz)



- A little difference when doubling the frequency can be ignored.  
(ex. C4 with frequency 262 Hz -> C5 with frequency 523 Hz)
- Volume control and octave control should take effect on both two modes with both tracks.
- Upon the **Reset** (BtnC), the volume should go back to its default level 3, and the octave should go back to its default level 2. Also, the demonstration song should start from the beginning.
- By switching the **Play/Pause** switch (SW 0) to “pause,” the piano should pause the demonstration song immediately and should play no tone. The piano should start/resume to play the song from where it paused by switching to “play.”  
The switch takes effect only on the DEMONSTRATE mode.
- By switching the **Muted** switch (SW 1) to 1, player can mute the piano. When muted, the key in the PLAY mode can still be pressed and the demonstration song in the DEMONSTRATE mode will keep playing. But you will hear no sound.  
The switch takes effect on both modes.
- By switching the **Slow Down** switch (SW 2) to 1, the piano should play the demonstration song two times slower than the normal speed. (Each note plays two times longer.) The piano should play the song at normal speed when the switch is switched to 0.  
The switch takes effect only on the DEMONSTRATE mode.
- By switching the **Mode** switch (SW 15), the player can change the piano mode: 0 for the PLAY mode and 1 for the DEMONSTRATE mode. If it is the first time switching to the DEMONSTRATE mode, the song should start from beginning. If it is NOT the first time switching to the DEMONSTRATE mode, the song should start from where it is paused (or switched to the PLAY mode) last time.

- LED 0~4 indicates the current volume level:

| Volume Level          | LED 4 | LED 3 | LED 2 | LED 1 | LED 0 |
|-----------------------|-------|-------|-------|-------|-------|
| Muted                 | ●     | ●     | ●     | ●     | ●     |
| Level 1               | ●     | ●     | ●     | ●     | ●     |
| Level 2               | ●     | ●     | ●     | ●     | ●     |
| Level 3               | ●     | ●     | ●     | ●     | ●     |
| Level 4               | ●     | ●     | ●     | ●     | ●     |
| Level 5 (the loudest) | ●     | ●     | ●     | ●     | ●     |

- LED 13~15 indicates the current octave level:

| Octave Level |        | LED 15 | LED 14 | LED 13 |
|--------------|--------|--------|--------|--------|
| Level 1      | Lower  | ●      | ●      | ●      |
| Level 2      | Normal | ●      | ●      | ●      |
| Level 3      | Higher | ●      | ●      | ●      |

- Display the melody pitch (C, D, E, F, G, A, and B) on the 7-segment display for both PLAY and DEMONSTRATE modes. Note that Sharp/flat notation need to be displayed also.  
For example:  
You should display ' - - G' when the first note of *Lightly row* is being played.  
You should display ' - - bB' when the third note of *Havana* is being played.  
You should display ' - - - ' when no key is pressed in the PLAY mode and when the demonstration song is paused in the DEMONSTRATE mode.  
You can use  to represent the sharp notation (#) and  to represent the flat notation (b).  
Note that when muted, the 7-segment display should still display the melody pitch.
- (Optional bonus) You can provide two demonstrate songs for the DEMONSTRATE mode. The piano will play one of them when the **Music** switch (SW 3) is 0, and the other when the switch is 1. Every time the Music switch is changed, the piano start playing the corresponding song from the beginning.
- Please refer to [demo video](#)

### I/O signal specification

|             |                         |                                       |
|-------------|-------------------------|---------------------------------------|
| BtnC        | Reset                   |                                       |
| BtnU        | Volume Up               |                                       |
| BtnD        | Volume Down             |                                       |
| BtnR        | Higher Octave           |                                       |
| BtnL        | Lower Octave            |                                       |
| SW 0        | Play / Pause            | 1: Play; 0: Pause                     |
| SW 1        | Mute / Normal           | 1: Mute; 0: Normal                    |
| SW 2        | Slow Down / Normal      | 1: Slow Down; 0: Normal               |
| SW 3        | Music (optional bonus)  | 0: the first song; 1: the second song |
| SW 15       | Mode                    | 0: PLAY mode; 1: DEMONSTRATE mode     |
| LED 0 ~ 4   | Volume Indicator        |                                       |
| LED 13 ~ 15 | Octave Indicator        |                                       |
| Pmod JB 1~6 | Pmod I2S                |                                       |
| 7-Segment   | Displaying Current Note |                                       |

### Attention

- You should hand in only one Verilog file, **lab8.v**.
- Finish the modules from the template, and integrate them in lab8.v. You don't need to put debounce, onepulse, keyboard\_decoder, and speaker\_control in the file you hand in. **Please do not integrate them in lab8.v.**
- You should also hand in your report as **lab8\_report\_StudentID.pdf** (e.g. **lab8\_report\_109062666.pdf**).
- Please do not hand in any compressed files, which will be considered as an incorrect format.
- You should be able to answer the questions of this lab from TA during the demo.
- You need to generate the bitstream before the demo.
- If you have any questions about the spec, feel free to ask on the EECLASS forum.

## Pick a Song

The design template plays the following music.

```
// --- Measure 1 ---
12'd0: toneR = `hg;    12'd1: toneR = `hg; // HG (half-beat)
12'd2: toneR = `hg;    12'd3: toneR = `hg;
12'd4: toneR = `hg;    12'd5: toneR = `hg;
12'd6: toneR = `hg;    12'd7: toneR = `hg;
12'd8: toneR = `he;    12'd9: toneR = `he; // HE (half-beat)
12'd10: toneR = `he;   12'd11: toneR = `he;
12'd12: toneR = `he;   12'd13: toneR = `he;
12'd14: toneR = `he;   12'd15: toneR = `sil; // (Short break)
12'd16: toneR = `he;   12'd17: toneR = `he; // HE (one-beat)
12'd18: toneR = `he;   12'd19: toneR = `he;
12'd20: toneR = `he;   12'd21: toneR = `he;
12'd22: toneR = `he;   12'd23: toneR = `he;
12'd24: toneR = `he;   12'd25: toneR = `he;
12'd26: toneR = `he;   12'd27: toneR = `he;
12'd28: toneR = `he;   12'd29: toneR = `he;
12'd30: toneR = `he;   12'd31: toneR = `he;
```

You can pick any song of the sample songs listed below. Pick two songs to design the bonus. The **Music** switch controls which one to play by your definition.

They are all 8 measures (小節) in length.

### 1. Lightly Row

### 2. Jingle Bells

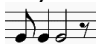
3. 貓咪大戰六周年廣告 (Note the bass clef  $\text{bass clef}$  )4. 熱愛 105 度的你 (Note the bass clef  $\text{bass clef}$  and Bb, Eb)5. Havana (Note the bass clef  $\text{bass clef}$  and Bb, Eb and some temporary F#)6. 夜に駆ける (Note the bass clef  $\text{bass clef}$  and Bb, Eb, Ab)

## 7. Mario Theme Music (Note some temporary sharp #, flat b, and natural n symbols)




## 8. Any (beautiful) songs you like!

Music of your choice/creation/arrangement should follow a few conditions below.

- Contain both melody and accompaniment parts.
- Not too short (with at least 8 measures).
- Contain at least 3 type of note. (like )

You should provide your score (樂譜) to TAs to see if it is good enough before working on the lab.

## Hints

- Trace code first. This lab won't be so hard if you realize how the template works.
- Here are two possible ways of thought to implement the PLAY mode with keyboard.
  - If a valid key is pressed, generate a pulse signal last for about  $2^{24}$  cycles (with 100MHz clock) for the corresponding note. ( $2^{24}$  is just for reference only, you can decide the length of the pulse signal by yourself)
  - Simply use key\_down from keyboard\_decoder.v to decide which key is pressed and which note to play.
- Remember that the buzzer/speaker uses 2's complement numbers for audio signals. When designing the volume level, choose the peak value to be some certain *val* and *-val*.
- If two consecutive notes have the same frequency, it may not be possible to tell when the second note starts. Therefore, in the template, one Quarter Note  is divided into 16 beats further, for the sophisticated note arrangement. You may use a short rest (silence) so that the 2 same-frequency notes can be separated by a short break. (Refer to music\_example.v).
- A note-to-frequency table is in the appendix for your reference.
- You can use multiple music modules or player modules if that helps.
- You can add or modify some modules in the template.
- We use [square waves](#). So, the music may not sound smooth.

- Since a quarter note is 16 beats, it may be tedious to enter all the notes one by one (8 measures consist of  $8 \times 4 \times 16 = 512$  beats in total). In that case, you may want to write an aid program like:

```

C:\Program Files\dotnet\dotnet.exe
12'd602: toneR = `c; 12'd603: toneR = `c;
12'd604: toneR = `c; 12'd605: toneR = `c;
12'd606: toneR = `c; 12'd607: toneR = `c;
[c]
12'd608: toneR = `c; 12'd609: toneR = `c;
12'd610: toneR = `c; 12'd611: toneR = `c;
12'd612: toneR = `c; 12'd613: toneR = `c;
12'd614: toneR = `c; 12'd615: toneR = `c;
[c]
12'd616: toneR = `c; 12'd617: toneR = `c;
12'd618: toneR = `c; 12'd619: toneR = `c;
12'd620: toneR = `c; 12'd621: toneR = `c;
12'd622: toneR = `c; 12'd623: toneR = `c;
[c]
12'd624: toneR = `c; 12'd625: toneR = `c;
12'd626: toneR = `c; 12'd627: toneR = `c;
12'd628: toneR = `c; 12'd629: toneR = `c;
12'd630: toneR = `c; 12'd631: toneR = `c;
[c]
12'd632: toneR = `c; 12'd633: toneR = `c;
12'd634: toneR = `c; 12'd635: toneR = `c;
12'd636: toneR = `c; 12'd637: toneR = `c;
12'd638: toneR = `c; 12'd639: toneR = `c;
[end]

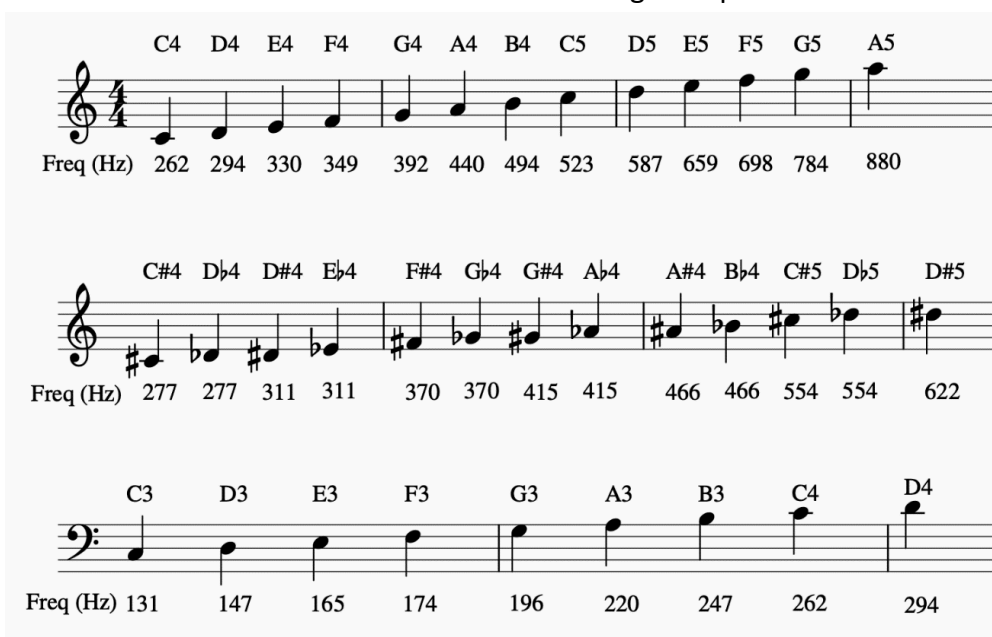
```

So that you can copy-n-paste them, saving a tremendous amount of time.

## Appendix

### Pitch-to-Frequency Table

The number after the notation indicates how high the pitch is.



You can use these frequencies in your code like that in `music_example.v`.

Or you can refer to [this page](#).

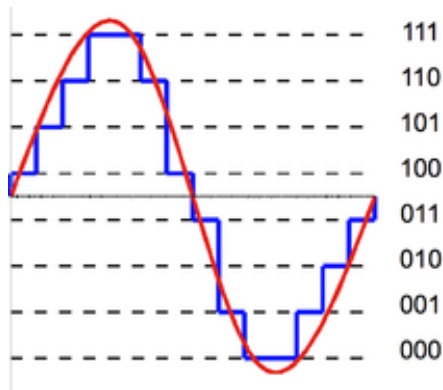
### Square Waves and Sine Waves

We use square waves to control the speaker, which inevitably results in a buzzing sound.

[Triangle waves](#) will do, but [sine waves](#) are usually the most natural.

Sine waves can be emulated in Verilog with a look-up table. But it results in some [quantization noises](#) (if no interpolation is involved) that make the sound terrible. (like the picture below, from Wikipedia)





However, there is an algorithm known as [CORDIC](#), or **Volder's algorithm**, a simple and efficient way to calculate trigonometric functions, even when using the FPGA. Refer to Wikipedia for more information. Vivado also provides the CORDIC IP in the IP Catalog. So, you may find it handy in this lab if you really cannot stand the square wave sound or if you are going to improve the audio effect in your final project. (You can also generate audio data in your PC and store it in the block memory of Basys3. But it will consume a lot of memory.)

- **Different Timbres (音色)**

Refer to [this video](#) to gain an insight of how timbres are made of. Actually, the frequency change (vibrations on violins), amplitude change (like the fading sound of pianos), and how the sound waves start can determine how we perceive the timbres. Even the ratio of overtones can differ on the same musical instrument when it plays different frequencies.