Lab 8: Music Player

Submission Due Dates:

Demo:	2022/12/13 17:20
Source Code:	2022/12/13 18:30
Report:	2022/12/18 23:59

Objective

Getting familiar with the audio peripheral and Pmod Connector.

Description

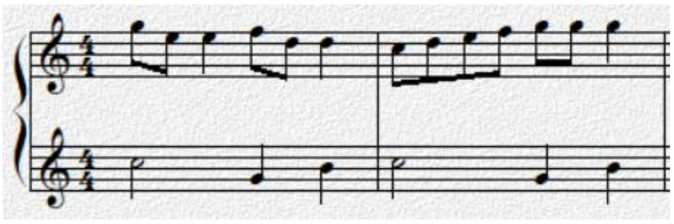
You plan to build a piano as a Christmas gift to a best friend who wants to be a pianist. To make this instrument easy to practice, you are designing an electronic piano with a computer keyboard as an instrument keyboard.

1. The piano has two modes:
   i. PLAY mode: In this mode, the user can play the piano using the keyboard and record the music notes.
   ii. DEMONSTRATE mode: In this mode, the piano can demonstrate the music notes that have been **recorded (when SW3=1)** or the **preset** music notes **(when SW3=0)**.
2. The piano should make no sound in the PLAY mode unless you press a key. The following table shows the key-note mapping.

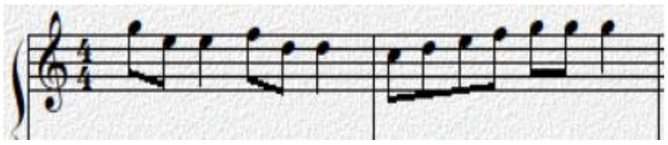
Key	a	s	d	F	g	h	j
Note	C (or Do)	D (or Re)	E (or Mi)	F (or Fa)	G (or Sol)	A (or La)	B (or Ti)

   If you think these notes are not enough to play a good song, feel free to add more notes with additional keys.
3. 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 should play the last note if multiple keys are pressed.
4. In the PLAY mode, the user can press the **record key** (keyboard **r**) and start the recording. The duration of the recording is **ten seconds**.
5. When pressing the record key (keyboard **r**), the previous recording will be cleared.
6. When recording, the LED 5 should be on (●). Otherwise, LED 5 should be off (●).
7. When recording, the **record key** (keyboard **r**) will not take effect again.
8. When recording, we expect no other operations will do except the keynotes.
9. In the DEMONSTRATE mode, the piano will play the preset demonstration song when the **Music** switch (SW 3) is 0 or the music recording when the switch is 1. Every time the Music switch is changed, the piano starts playing from the beginning.
10. You can pick any song from the sample sheets below for the demonstration song.
11. In the DEMONSTRATE mode, the piano will **repeatedly** play the demonstration song (starting from the beginning again when reaching the end) and will NOT react to any key pressed on the keyboard.
12. The piano should support two tracks.
    i. In the PLAY mode and the music recording, two tracks play the same note.
    ii. In the DEMONSTRATE mode for the preset song, one track plays the melody part, and the other plays the accompaniment part.

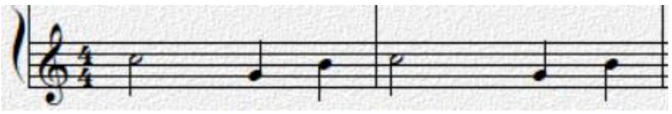
E.g., if these are two measures of the preset demonstration song:



The track for the melody part:



The track for the accompaniment part:



13. The piano should support five distinguishable levels of volume, controlled by **Volume Up** (BtnU) and **Volume Down** (BtnD). The default volume level is 3. You decide how loud these five levels are. But note that, at the lowest level, the sound should still be able to be heard. Pressing **Volume Up** at level 5 or pressing **Volume Down** at level 1 takes no effect.
14. The piano should support three levels of octaves, controlled by the **Higher Octave** (BtnR) and **Lower Octave** (BtnL). The default octave level is 2. Pressing the **Higher Octave** at level 3 or pressing the **Lower Octave** at level 1 takes no effect.
    Raising a note an octave higher can be done by doubling that note's frequency
    (e.g., A4 with frequency 440 Hz -> A5 with frequency 880 Hz).
15. A slight mismatch when doubling the frequency can be ignored
    (e.g., C4 with frequency 262 Hz -> C5 with frequency 523 Hz).
16. The volume control and octave control should take effect for both tracks at the same time in the two modes.
17. 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, and the recording should be cleared.
18. By switching the **Play/Pause** switch to “**pause**” (SW0=0), the piano should pause the demonstration song immediately and should play no tone. The piano should start/resume playing the song from where it paused by switching to “**play**” (SW0=1).
    The switch takes effect only in the DEMONSTRATE mode.
19. The user can mute the piano by switching the **Muted** switch (SW 1) to 1.
20. 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 in both modes.
21. The record key (keyboard r) will take effect when muted.
22. By switching the **Slow Down** switch (SW 2) to 1, the piano should play the demonstration song two times slower than the normal speed (i.e., each note plays two times longer). The piano should play the song at normal speed when the switch is 0.
    The switch takes effect only in the DEMONSTRATE mode.
23. The piano is in the PLAY mode when the **Mode** switch (SW 15) is 0 and in the DEMONSTRATE mode when the switch is 1. If it is the first time switching to the DEMONSTRATE mode, the

song should start from the beginning. Otherwise, the song should start from where it was paused (or switched to the PLAY mode) last time.

24. LED0~LED4 indicate 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)	●	●	●	●	●

25. LED13~LED15 indicate the current octave level:

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

26. 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 notations need to be displayed as well.
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 pitches.

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	0: the preset demonstration song;

		1: the recorded music notes
SW 15	Mode	0: PLAY mode; 1: DEMONSTRATE mode
LED 0 ~ 4	Volume Indicator	
LED 5	Music Recording	
LED 13 ~ 15	Octave Indicator	
Pmod JB 1~6	Pmod I2S2	
7-Segment	Displaying Current Note	

DEMO: https://reurl.cc/deLX2k

Questions and Discussion

Please answer the following questions in your report.

A. If we want to play the song in the backward sequence (i.e., from the tail to the head), how would you modify your design of Lab 8?

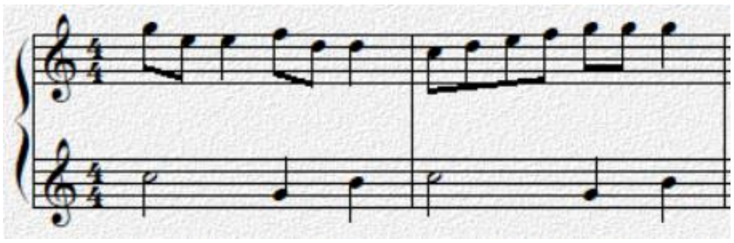
B. What happens when we change
```
next_ibeat = (ibeat + 1 < LEN) ? (ibeat + 1) : LEN-1;
```
to
```
next_ibeat = (ibeat + 2 < LEN) ? (ibeat + 2) : LEN-1;
```
?

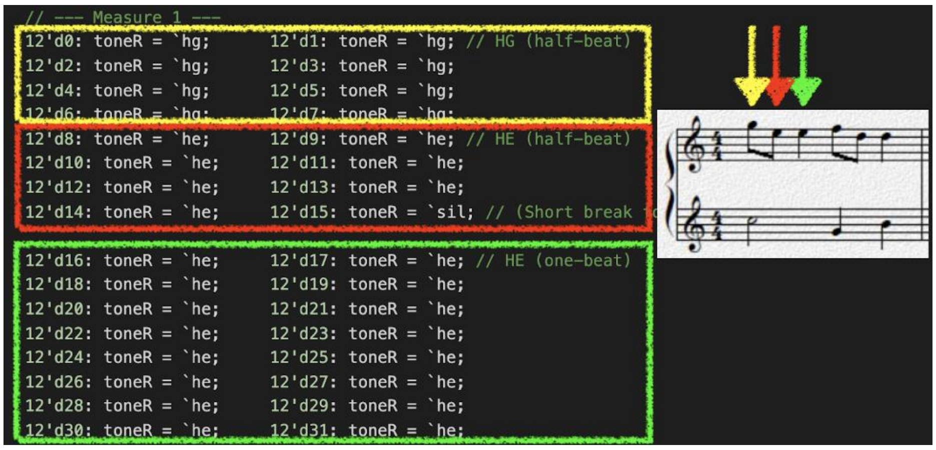
Attention

- ✓ You should hand in only one source file, lab8.v.
- ✓ Finish the modules from the template, and integrate them in lab8.v. If you have several modules in your design, integrate them in lab8.v.
- ✓ DO NOT include the clock_divider, debounce, one-pulse, keyboard_decoder modules and speaker_control into lab8.v.
- ✓ You should also hand in your report as **lab8_report_StudentID.pdf** (i.e., lab8_report_110062666.pdf).
- ✓ DO NOT hand in any compressed ZIP files, which will be considered an incorrect format.
- ✓ You should be able to answer the questions for this lab from TA during the demo.
- ✓ You need to prepare the bitstream files before the lab demo to make the demo process smooth.
- ✓ Feel free to ask questions about the specification on the EECLASS forum.

Pick a Song

The design template plays the following music.



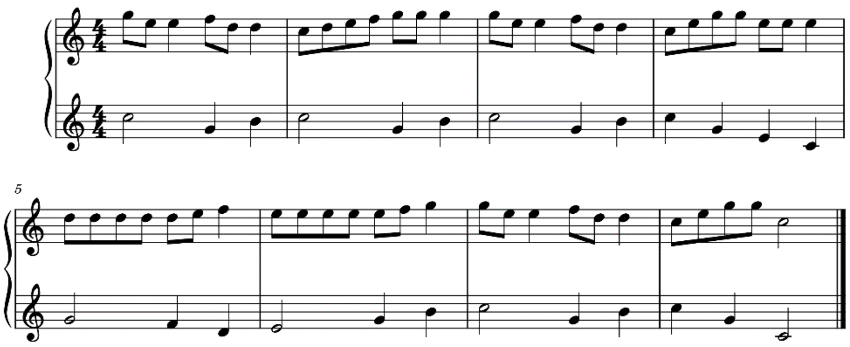


```
// --- Measure 1 ---
12'd0: toneR = `hg;      12'd1: toneR = `hg; // HG (half-beat)
12'd2: toneR = `hg;      12'd3: toneR = `hg;
12'd4: toneR = `hg;      12'd5: toneR = `hg;
12'd6: toneR = `hg;      12'd7: toneR = `hg;
12'd8: toneR = `he;      12'd9: toneR = `he; // HE (half-beat)
12'd10: toneR = `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;
```

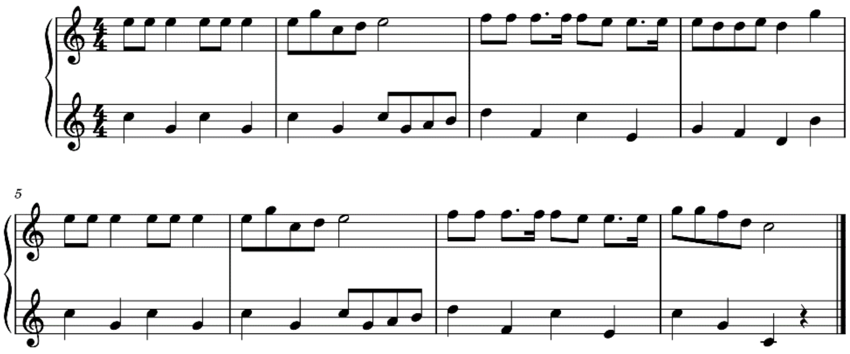
You can pick any of the following songs as the preset demonstration song.

They are all eight measures (小節) in length.

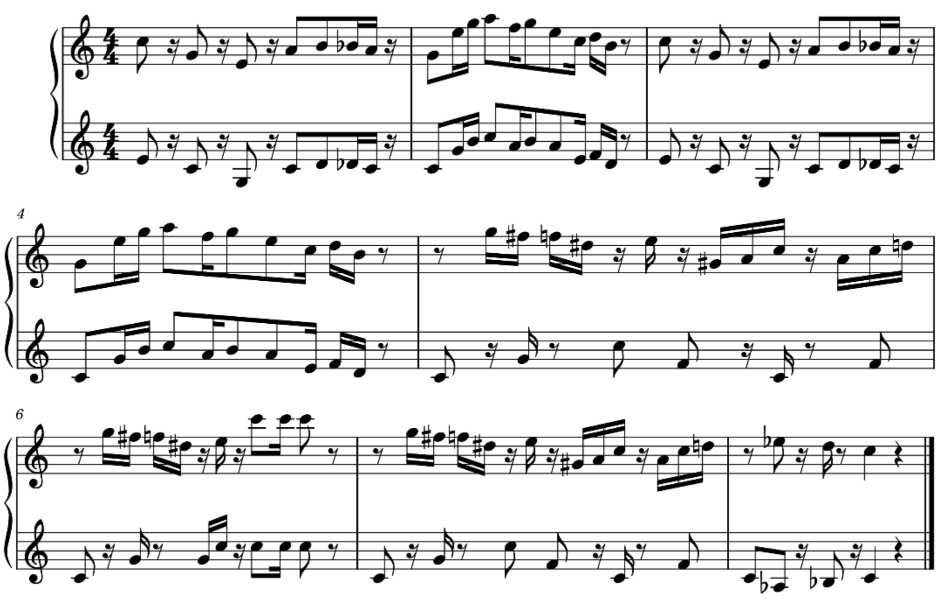
1. Lightly Row



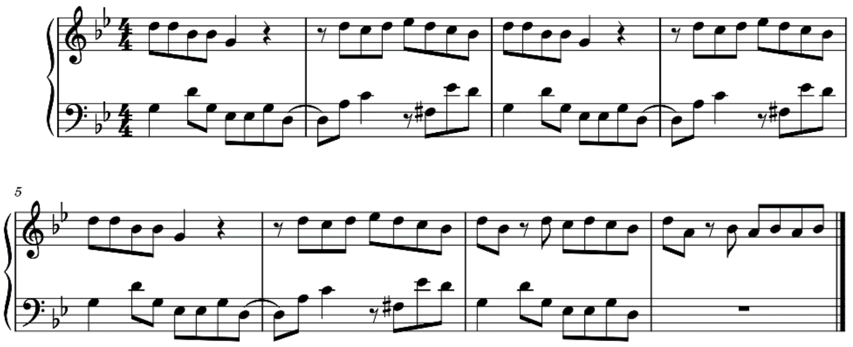
2. Jingle Bells

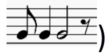


3. Mario Theme Music (Note that there are several temporary sharp ♯, flat ♭, and natural ♮ symbols)



4. Havana (Be careful of the bass clef (𝄢), B♭, E♭, and some temporary F♯)

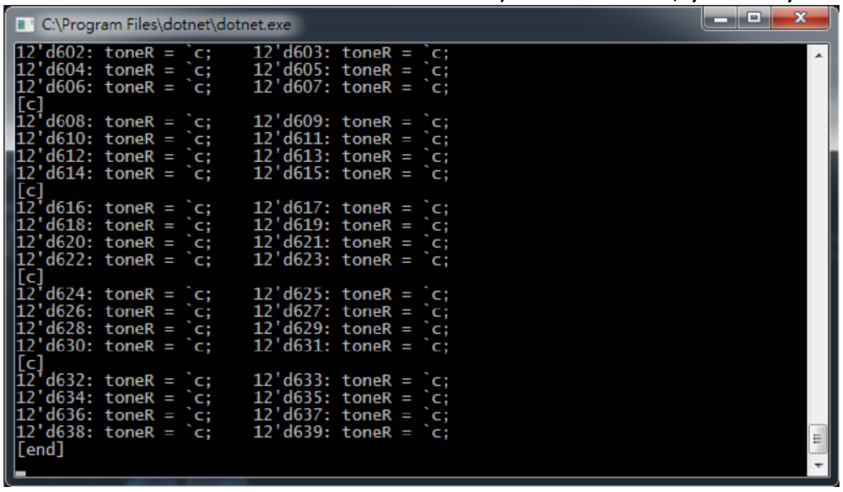


5. Any (beautiful) songs you like!
   Music of your choice/creation/arrangement should follow a few conditions below.
   a. It must consist of distinguishable melody and accompaniment parts.
   b. It must include at least eight measures.
   c. It must include at least three types of notes (e.g., ).
   You MUST provide your score (樂譜) to TAs to see if it is good enough before working on the lab. Otherwise, you may get a zero score!

Hints

- Trace the sample code first.
- Here are two possible ways to implement the PLAY mode with the keyboard.
  1. If a valid key is pressed, generate a pulse signal lasting for about 2^{24} cycles (with the 100MHz clock) for the corresponding note. (2^{24} is just for reference only, you can decide the length of the pulse signal by yourself)
  2. Use the key_down from the keyboard_decoder.v to decide which key is pressed and which note to play.
- Remember that the buzzer/speaker uses two'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) to separate the two same-frequency notes (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*4*16 = 512 beats in total!). In that case, you may write an aid program as follows:



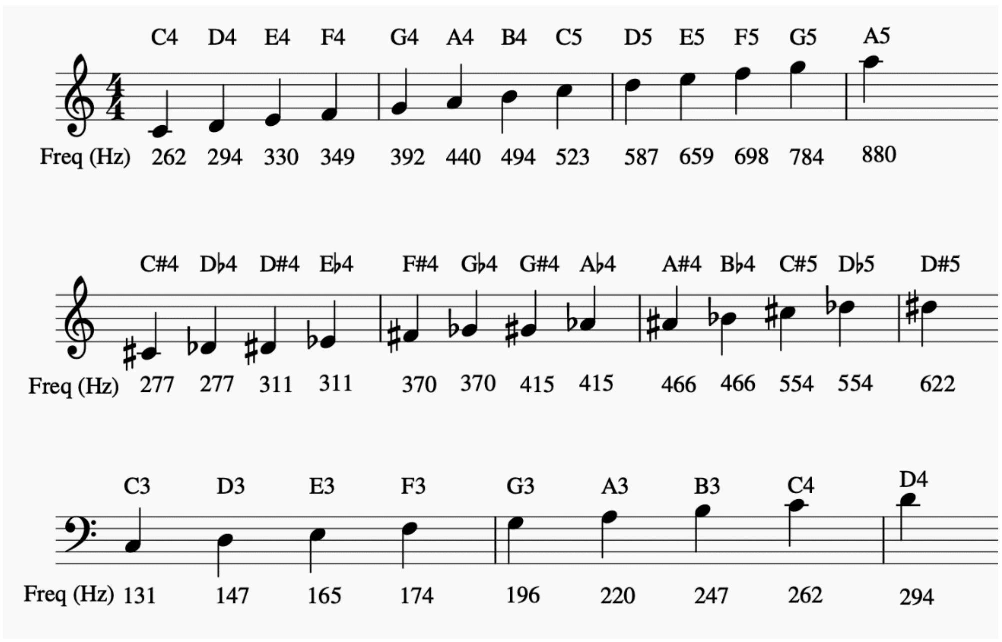
```
12'd602: toneR = `c;     12'd603: toneR = `c;
12'd604: toneR = `c;     12'd605: toneR = `c;
12'd606: toneR = `c;     12'd607: toneR = `c;
[c]
12'd608: toneR = `c;     12'd609: toneR = `c;
12'd610: toneR = `c;     12'd611: toneR = `c;
12'd612: toneR = `c;     12'd613: toneR = `c;
12'd614: toneR = `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**

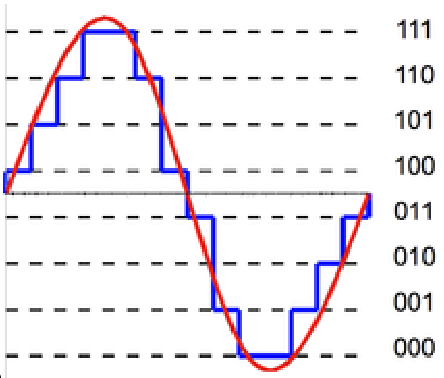
  The number after the notation indicates how high the pitch is.


C4 D4 E4 F4 G4 A4 B4 C5 D5 E5 F5 G5 A5
Freq (Hz) 262 294 330 349 392 440 494 523 587 659 698 784 880
C#4 D♭4 D#4 E♭4 F#4 G♭4 G#4 A♭4 A#4 B♭4 C#5 D♭5 D#5
Freq (Hz) 277 277 311 311 370 370 415 415 466 466 554 554 622
C3 D3 E3 F3 G3 A3 B3 C4 D4
Freq (Hz) 131 147 165 174 196 220 247 262 294

You can use these frequencies in your code. Refer to music_example.v or 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 on 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 into 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.