Design document

Name: Yixi Rao

Uid: u6826541

Date: 09/04/2020

Overview: I made synthesizer which can produce a musical sound of a wave I designed. And the merged wave is made up of a square wave, a sawtooth wave, a triangle wave, another square wave and finally is an additive synthesis wave respectively. About the square wave, it has an amplitude of 0xC350 and it has 40 dots in one period, the sawtooth with an amplitude of 0xD2F0 and 120 dots in one period, the triangle wave starting value is 0x6978 and it will first decline and then rise, so the amplitude is 54000 and 90 dots in one period. And about another square wave, which has an amplitude of 0x2710 and has 90 dots in one period (the first four waves are in figure 1). The last one is an additive synthesis wave that waveform is the addition of a triangle wave and a sawtooth wave, the triangle wave with an amplitude of 0x8CA0 and the sawtooth wave with an amplitude of 0x4E20 shares the same frequency and same dots in one period (400), and this waveform shape is like a check mark or a tick mark (figure 2). The reason why I designed this waveform is that I want to hear what is the mixed version of these their common waveform and add an additive synthesis wave to make the pitch of the whole sound deeper.

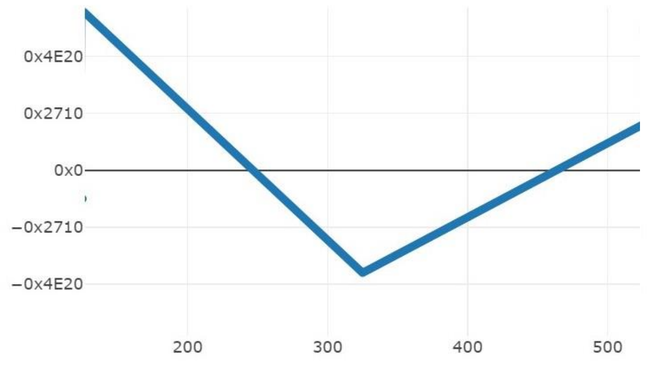

Figure 1

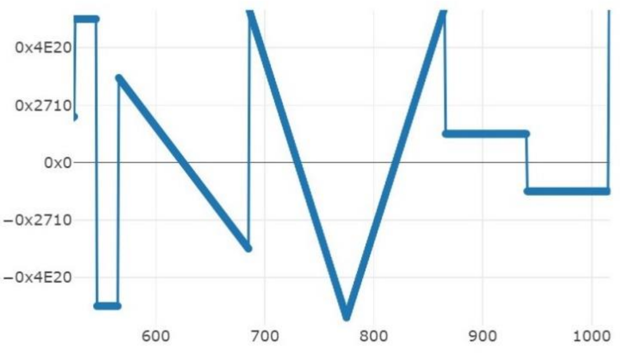

Figure 2

Implementation: my design decision of structuring and organizing the whole program is dividing the whole structure and code into different parts or sections, and each section will finish a specific wave. The reasons why I chose this structure are that it can make the code more distinct and intelligible because for each section there is a label on it make easier for the reader to read and understand, easier for me to debug as well, the other reason is that one section of code will not confuse the other section of the code and the same time the different section can be connected like a chain.

Algorithm: All of my algorithms of creating a different waveform have strong similarity, so the basic algorithm is like there are two while loops inside a big loop, the first while loop will do some operations and the other one will do the opposite operations. No matter what waveform it is except sawtooth wave, it will have one inflection point and two different but symmetric lines, so the two while loops will generate two different lines respectively and easily create that inflection point and it will finally create a period of a wave, so that is the reason why I use this two while loop do the opposite operations in this algorithm. The advantage of this algorithm is that it can fully present the logic of this program and not confuse the assembly code but the disadvantage is that it has a lot of redundant and repeated codes. The detailed operation and the assembly code are different according to a different waveform, but there are all changing the value in r0, such as for the triangle wave and sawtooth wave the value in r0 must be descending or ascending so it can create a line as the time passes, however, the square wave has to keep a constant value because its slope must be zero. For the special case sawtooth wave, the design is that only one while loop will fit because the sawtooth wave only has one line in a period so just use one loop can solve this.

Additive synthesis wave: My additive synthesis wave is a combination of a triangle wave and a sawtooth wave, there are many choices of implementing this additive synthesis way such as two while loop for creating this two different waveform and when you create one value of one wave than branch to another one to create another waveform value and add them together, and finally go the sound playing section to read that value in r0 and play, another way is that actually creating an additive synthesis value can just happen in one while loop, so which means that we can do the operations for the sawtooth wave and a triangle wave at the same loop and add it together to play this signal no longer need to have an extra sound playing section. The method I chose is the second one because we can easily discover that all the waveform has two different but symmetric shape, therefore we can use this characteristic to collect the fragment of the wave that they have the same feature such as descending or ascending, and then add them into the same while loop, it is more space-efficient method when it compared with the first method because the first method principle is just combined two normal sawtooth waves and triangle wave with some modification into a big loop, so it totally has four small while loops while the second one just has two while loops because all the waveform at most have two diverse characteristics so it is space-saving, and if we want to combine or add more than two waveforms, the second method always has two while loops while the number of first one while loop will grow linearly with the number of waveform. The most tricky or confusing part of this assembly code maybe is the how to deal with the different and changing coordinate of these two waveforms, the fact is that we only have one register r0 that is going to be passed to the BSP_AUDIO_OUT_Play_Sample function, but we have two waveforms in one while loop, so my solution to solve this situation is that copy the other waveform value to a new register to prevent losing data.

Reflection: To be honest, I think my assembly program is still very repeated and some codes can be classified as the same category as they have the same structure but with different parameter, therefore function can be used in my program to replace the redundant codes, such as all the calculation part can be written in a "next point function" which can calculate the next dot coordinate. If I learn more about how to use the function and know more about the function with parameter and return value, I can implement the last type of waveform that I did not create in my program, which is the sin wave, it is very hard to generate this waveform without using function because the derivative is changing over time, however, other waveforms have a constant- derivative, so we know the offset of the dot is eternally unchanging, and also the content in the while loop cannot be changed over time therefore it is hard to change the constant offset in the while loop. But using function will solve this problem because function can change each dot offset value, by writing a formula into the function and calculate the sin value offset of the coordinate.

Writing and designing this program, I learn a lot from it, for example I learn how to use the bl instruction and function, the value in some registers will change when call back from a function as well, but most important thing I learn is the assembly code structure and layout such as I understand if we want to solve a problem we should design a structure first and then write the code, so choosing a structure is very useful and can help us solve the problem faster.