16-Step Sequencer and Synthesizer

Zhuowen Lin, 11510818, *Information Engineering, SUSTech*

Jinyue Guo, 11510478, *Information Engineering, SUSTech*

I. Introduction

MUSIC sequencer is a device that can edit, record or play back music notes. It is one of the most fundamental components in music creation or live show. One of the commonly-seen step sequencers is shown in Fig. 1.

Fig. 1. Real world 16-step sequencer.

The array of square touch pads at the top right corner is the core of step sequencer. The horizontal direction of these touch pads corresponds to time, which means that the time passes by the touch pads column by column. The vertical direction of these touch pads corresponds to different tones. When the time passes by a specific column, the sequencer plays out the tones corresponding to the lighted up pads in that column.

Software music sequencer can be played in several websites, http://tonematrix.audiotool.com/ is one of which. In the given website, the sequencer has 16 steps and 16 different tones. Our project is to implement a hardware version of this software 16-step sequencer. Therefore, it would be recommended to play and enjoy the software sequencer in that website before reading the following parts (VPN with overseas server is needed to visit this website).

II. Inputs and Outputs

The introduction of the overall inputs and outputs of our step sequencer and synthesizer may contribute to a brief comprehension about what this project does. Therefore, they will be introduced first in this report.

The inputs are listed as follows:

1. Switches. All 16 switches on the Nexys 4 DDR board are used. Each of the 16 switches corresponds to a music note at that specific time, or namely, step. If the switch is on, the corresponding step is activated, thus the corresponding music note on that step will be played out when the time passes by.
2. Buttons.
   a) Central button BTNC: Play and Pause. The main function of this button is to control the play and pause of our step sequencer.
   b) Left button BTNL: Reset. If this button is pressed, all the assigned notes would be clear.
   c) Up button BTNU: Switch Frequency. 16 different frequencies ranging from a to g3 are preset into the hardware. By pressing this button, the preset frequencies would appear one by one and the user can choose the music note they want.
   d) Right button BTNR: Save. After choosing the wanted frequency by pressing BTNU, pressing BTNR can assign and save the current frequency to all the switches that is in the on position.

The outputs are listed as follows:

1. LEDs. There are 16 LEDs above the 16 switches. If a LED lights up, it means that a music note is currently playing on that particular switch.
2. Seven Segment Display. The seven segment display on the board is used to show which of the 16 preset frequencies is now being worked with. Therefore, hexadecimal number from 0 to F will be used to display.
3. Speaker. There is a 3.5 mm mono-channel audio output port on the Nexys 4 DDR board. Sound waves at different frequencies generated by the board will be output to this port and then be played out.

Zhuowen Lin is with the Information Engineering major, Southern University of Science and Technology, Shenzhen, Guangdong, China (e-mail: 11510818@mail.sustc.edu.cn).

Jinyue Guo is with the Information Engineering major, Southern University of Science and Technology, Shenzhen, Guangdong, China (e-mail: 11510478@mail.sustc.edu.cn).

III. Components of Sequencer Circuit

A. Button Toggle – Play and Pause

The first main component of the whole program deals with the function of central button BTNC – play and pause. When combining play and pause functions into one button, one problem is met. We cannot touch the button for such a short time to trigger the button only once. Even though we touch and leave the button very rapidly, it actually gives out multiple trigger signals, making the step sequencer switching between play and pause.

In order to solve this problem, FSM must be designed to handle multiple trigger signals. The FSM is shown in Fig. 2.

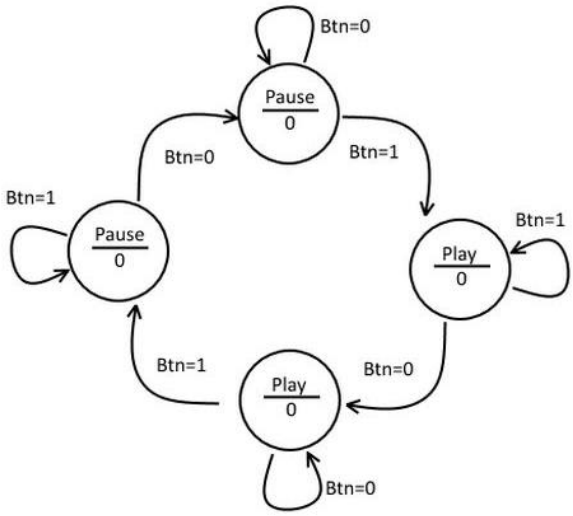

Fig. 2. Finite state machine of button toggle.

The comprehension of this FSM starts from the top state. When the FSM is in the top state and the button is pressed to give the trigger signal (Btn=1), it will transfer into the right state first and maintain at the right state if the signal is still being triggered. Once the button is released, the trigger signal is cleared (Btn=0), the FSM will transfer from the right state to the bottom state. In this state, the machine is still playing music notes but it will wait for the next trigger signal. Once the button is again pressed to give a sign of pause, the FSM will transfer to the left state and give the music a pause. If the button is released, the FSM will again be in the top state, waiting the next trigger signal to give a sign of play.
By doing this, we can toggle between two states in FSM exactly once when the central button is pressed.

B. Button Debouncer

Because the buttons on the Nexys 4 DDR board are not perfect, sometimes when we press and hold the button, it will not stay together all the time until the button is released, giving some random short-time switching between 0 and 1. The button contacts will tend to bounce between 0 and 1 for several times in a short time scale before giving out stable 1 signal. This phenomenon will cause the FSM introduced in previous part bouncer between states, which is not preferable.

We use a button debouncing program developed by Scott Larson in "Digi-key" website to handle this problem.

The circuit of this program is shown in Fig. 3.

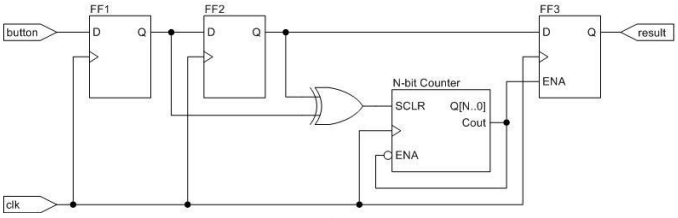

Fig. 3. Circuit of button debouncer.

FF1 and FF2 are two D flip-flops. The circuit continuously stores the button's logic signal into these two D flip-flops. The N-bit counter is responsible for timing. If the button bounces, the values stored in FF1 and FF2 will become different, after going through the XOR gate, the value in N-bit counter will be cleared to 0 and the counter will restart its counting. If the button does not bounce for a certain time period, the values stored in FF1 and FF2 will be the same in this time period, making the counter increases its value. If the counter value reaches a specified threshold, FF3 will be enabled and give out the final debounced result. The circuit will remain in this state until FF1 and FF2 store different values.

In the program, the threshold value of the counter is set to 10.5 ms. This is because, according to the original author of the program in "Digi-key", most switches reach a stable logic level within 10 ms of the actuation.

C. Clock Dividor

Dividing the 100 MHz clock signal into slower clock, just as what we have done in our lab, enables us to set the beat-per-minute (BPM) of the step sequencer.

D. T Flip-Flop Counter

4-bit T flip-flops are used in the program to act as a role as 4-bit counters. The circuit of the 4-bit counter is shown in Fig. 4.

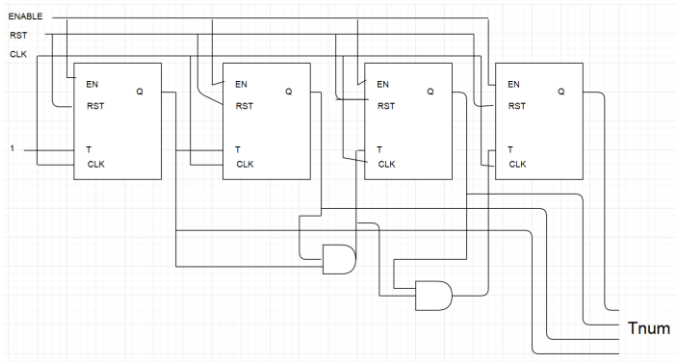

Fig. 4. Circuit of T flip-flop counter.

As can be seen from the circuit, the least significant bit will be toggled each time, while the higher order bits will only be toggled when all of the bits below them equal to 1.

The 4-bit counters count from 0 to F and then rotate back to 0, thus they are used to keep track of something. First, they can be used to keep track of the switches. 16 switches correspond to the 16 values in 4-bit counter, so the speaker output can know which switch in on and play out its corresponding music note. The increasing speed of the counter is the BPM we set in clock dividor.

Second, the 4-bit counter can also be used to keep track of the tones. 16 preset tone frequencies correspond to the 16

values in 4-bit counter, so the hardware board knows which tone frequency is now being chosen by the user. The value of this 4-bit counter is displayed in seven segment display to let users know they are choosing which tone frequency.

E. Seven Segment Display

We use the program in course lab 3 to do the hexadecimal number seven segment display. As just mentioned, the seven segment display shows the value of the 4-bit counter keeping track of tones to let users know they are choosing which tone frequency.

F. D Flip-Flop

16 D flip-flops are used in the program to remember the chosen tones of 16 switches, respectively.

IV. Components of Synthesizer Circuit

The aim of audio synthesizer is to generate the output waveform of audio signal, it has to be a frequency-adjustable waveform generator, which takes input tone controlling signal and outputs the corresponding waveform. Also, according to the manual of Nexys 4 DDR board, we applied a Pulse Width Modulator (PWM) in order to implement the DAC process.

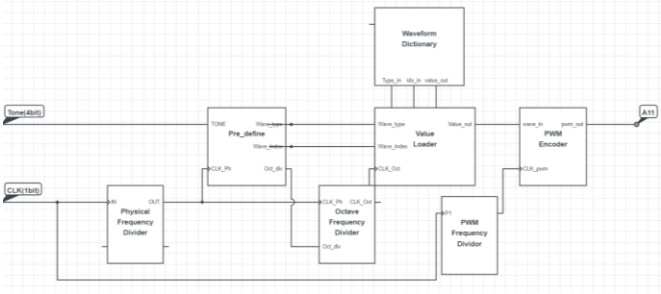

Fig. 5. Data flow of Audio Synthesizer

A. Physical Frequency Divider

This divider takes the board CLK as input, and reduces the frequency into hearable frequencies. The dividing coefficient is counter_phys_max = 46. It can turn the output waveform matches their standard frequencies as below:

A_4	440.00
$A^{\#}_4/B^{b}_4$	466.16
B_4	493.88
C_5	523.25
$C^{\#}_5/D^{b}_5$	554.37
D_5	587.33
$D^{\#}_5/E^{b}_5$	622.25
E_5	659.25
F_5	698.46
$F^{\#}_5/G^{b}_5$	739.99
G_5	783.99
$G^{\#}_5/A^{b}_5$	830.61
A_5	880.00

Table 1. Standard frequency of musical tones/

B. Pre-definder

It determines three coefficients, according to the input 4-bit tone signal.

There are five waveforms saved in the waveform dictionary, which are all 512-length arrays. We chose the pentatonic tones as our waveforms, namely C, D, E, G and A.

Signal *wave* determines which of these five arrays will be loaded.

Signal *max_index* determines the corresponding loading range. Since the 512-length array is not a whole period, we only takes the first period as our output value. *Max_index* is different for all five dictionaries.

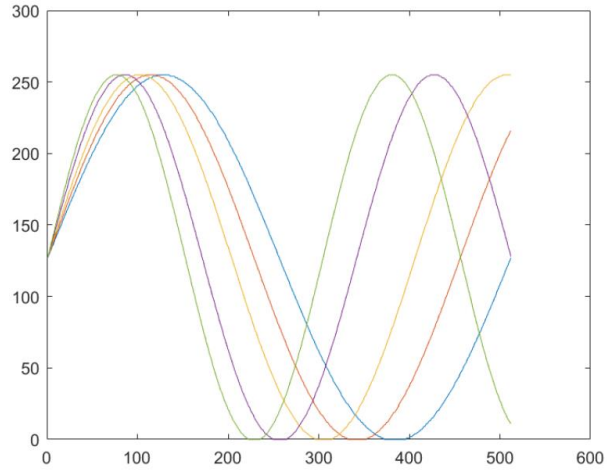

Fig. 6. Five arrays in the waveform dictionary.

Tone	Max_index
C	511
D	455
E	407
G	342
A	305

Table 2. max_index of five dictionaries

Signal *counter_div_max* determines the octave frequency divider coefficient. Since C5 has the half frequency of C6, and C4 has the half frequency of C5, we can generate tones of different octaves by dividing the frequency of C6 into 1/2 or 1/4

C. Octave Frequency Divider

It takes the physical CLK signal as input CLK, and further divides it by the octave divider coefficient.

D. Output Value loader

It will load the value according to *wave* and *max_index* generated by *B.* The loading frequency is the octave-divided frequency generated by *C.*

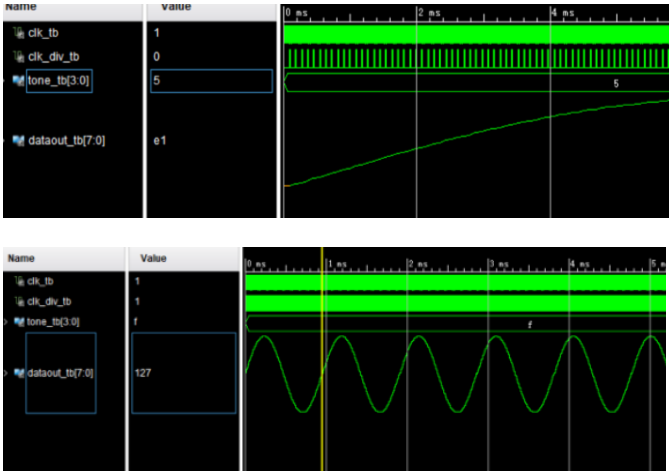

Fig. 7. Output waveform of A4 (above) and A6 (below), A6 is four times of A4.

E. PWM Frequency Divider

Determines the frequency of PWM encoder. This is the actual sampling frequency, so generally it should be greater than 44.1 kHz in order to cover the frequency range of hearing.

Even though the board manual says that PWM frequency should be as high as possible, we find out that the output volume will be extremely low if the PWM frequency is too high. This is due to the property of the LPF circuit before the output jack on the board.

The final coefficient is 2048, which makes the sampling frequency equals to 48.828 kHz. We think it is the best choice to make the output signal achieves the best sound quality as well as acceptable volume.

F. PWM Encoder

Generates the binary output waveform according to the input 8-bit value.

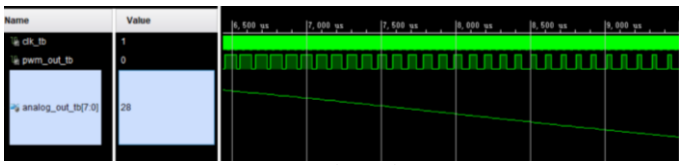

Fig. 8: Output waveform of PWM encoder

In the post-implementation timing synthesis, we have found that the 8-bit waveform has some noise, which might because of the flip-flop error in *D.* But the PWM encoder has the function of a down sampler, thus it could some-how reduce the output noise.

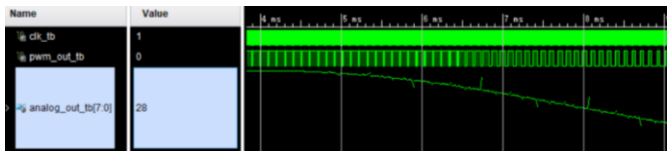

Fig. 9: Post-implement timing synthesis of PWM encoder. The transient error of analog output has been reduced.

The output signal of PWM encoder is directly connected to the pin of speaker.

REFERENCES

[1] Frequencies for equal-tempered scale, A4 = 440 Hz http://pages.mtu.edu/~suits/notefreqs.html

[2] Nexys 4 DDR Reference Manual: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual

[3] PWM Audio Tutorial using a FPGA https://www.youtube.com/watch?v=4byHVqXD-UI

[4] Class Slides