Melody Generator

Victor Wszedybyl and Kevin Kiliman
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: vcwszedybyl@oakland.edu, kpkiliman@oakland.edu

***Abstract*—This project is a recreation of a melody generator, using a Nexys 4 DDR board and an audio speaker. It serves the purpose of allowing a user to create sound melodies. The switches function as keys on a keyboard and correspond to individual notes that can be modified in the program. The tone of each note can be changed to create a desired melody. These melodies are then played through the attached speaker.**

I. Introduction

This project will cover the necessary architecture to create a melody generator on a Nexys 4 DDR board.

The motivation behind the project was to be able to create different sounds using switches on the nexys board and output them through a connected speaker to allow the user to create their own melodies. The project will incorporate the class topics of pulse width modulation (PWM) signals and outputting a signal through a speaker. This project would allow anyone with a Nexys 4 board and a simple speaker to make any sound melodies of their choosing and modify with any of the available switches and buttons.

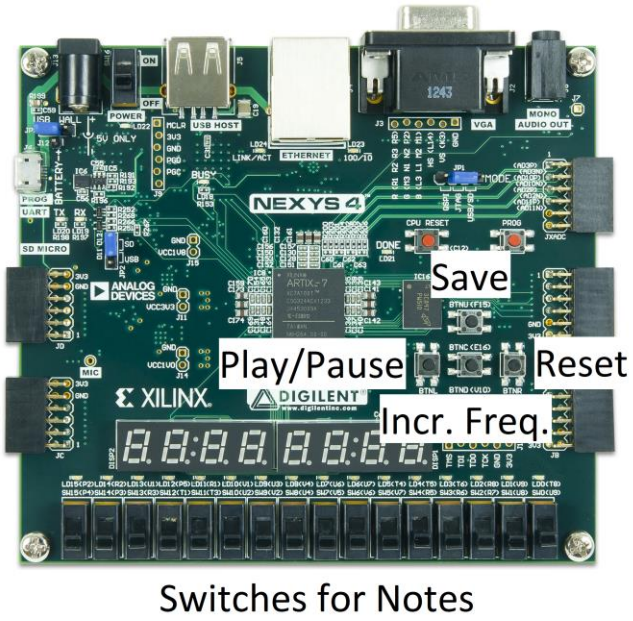
II. Methodology

There are 16 available switches on the Nexys 4 board that have the potential to play a note. There are also 4 buttons being used on the board. These buttons are Save, Play/Pause, Increment Frequency, and Reset. Turning a switch to the "On" position will activate it and then allow the user to save a note to it once a frequency is chosen and the user hits the save button. The user can program a total of 16 different notes but if a note is never programmed it is skipped over. This allows our note generator to have a melody duration anywhere from 1 to 16 notes. An LED above the switch is lit up as the Nexys4 plays that note. This is a visual cue so the user can tell what switch is being played at what time. The attached speaker is used to output the melody saved. All functional coding is done in VHDL on the Nexys board.

A. Play/Pause Button FSM

The play/pause button is designed to toggle between two states. In order to do this, a finite state machine (FSM) is used. The function of the FSM is to not allow another state transition until the button is released and then pressed again. A diagram representing this FSM is shown below.

Each possible output in the FSM has two states. Beginning at the top of the FSM diagram, the circuit is in the pause state. The circuit goes into the play state when the button is pressed and will remain in this state until the button is released. After the button is released, the circuit is ready for the button to be pressed again to transition to the pause state. The same thing occurs when waiting for the button to be released before becoming ready to switch back to the play state.



B. Clock Divider

The clock divider in the circuit is used to convert the base clock MHz frequency of the Nexys 4 DDR to a smaller frequency so a user has time to hear that sound. This is used to set the tempo of the melody generator. This has the possibility to be modified to change the BPM or beats per minute.

C. Counter

This project incorporated two counters. One is three bits and the other is four. The three-bit counter is used to cycle between possible notes, and the four bit counter plays through the 16 switches on the nexys4 board. Using flip-flops, we can cycle through the numbers to output a three or four-bit binary number. When the counter enable is high, it

will increase the number by 1 on the rising edge of clock. Each flip-flop represents a single bit being stored. Every time the counter is supposed to increment, the flip-flop that represents the least significant bit will toggle. The rest of the flip-flops will toggle once all of the bits below them are equal to one. When the counter reaches the highest possible number, it will reset back to zero when incremented next.

*D. **Debouncer***

Due to the nature of push buttons on the Nexys 4 DDR FPGA board, a button debouncer is required for two of our buttons (Play/Pause and Increment) When pressed, an ideal button's contacts would touch together exactly once, and stay together until the button is released. However, real buttons like the ones found on the Nexys 4 DDR, have contacts that have a tendency to "bounce" multiple times. When a button bounces the contacts will come in and out of contact several times before staying together. In context of this project, this bouncing could lead to a play/pause button which switches several times between the two states when the button is pressed, or incrementing the frequency more than once when it is not desired. In order to avoid this, the debounce circuit connected to our four buttons will wait a certain amount of time after the input is pushed to check if it stays in a constant state.

*E. **Previous State Checker***

As long as they are enabled, the counters in this project will increment along with a predetermined clock frequency. For some functions, such as pressing the button to increment the frequency of the current tone, it would be ideal to increment a counter only one time with each button press. To do this, converting the input to a pulse with a duration of one clock cycle is required. This is the function of the pulse generator. When the input of the pulse generator goes high, its output goes high for exactly one clock cycle. This is accomplished by checking the current input for a given rising edge of the clock, as well as checking the previous input for the last rising edge of the clock. The output is high when the current input is high and the previous input is low. Otherwise, the output is low. A behavioral model is used to do this in VHDL. This is done so when pressing the button, it will only increment to one higher frequency.

PWM Tone Generator

This function of this component is to generate a square wave based on a 4-bit input. This is done by dividing the input clock signal by a specific number to achieve the desired frequency. This frequency then determines when the output is toggled on and off. For this project, there are seven predetermined frequencies which represent the notes from A to G on a musical scale. The frequencies for these tones range from 405 Hz to 862 Hz to represent notes G# to A. If the input is 0000 in binary, then no square wave is played.

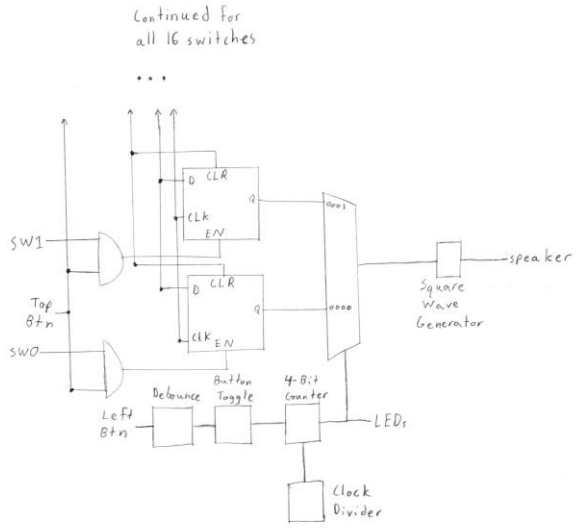
III. EXPERIMENTAL SETUP

The hardware used in this project was included a Nexys 4 DDR board, a speaker, and cables to deliver a signal from the board to the speaker. The software used was Vivado and all of the code was created in VHDL.

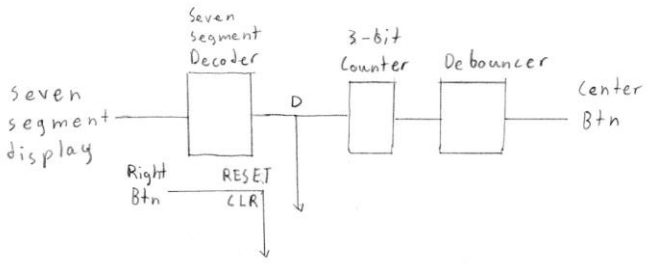
The final results were as expected. Each button, switch, and LED worked as it should. The desired sound output through the speaker also matched the user input.

Note	Frequency	7-Segment Number
$G\#_4$	405Hz	1
$A\#_4$	473Hz	2
C_5	526Hz	3
D_5	588Hz	4
F_5	684Hz	5
$F\#_5$	746Hz	6
A_5	862Hz	7

This is a chart matching the number being displayed on the seven-segment Display. This allows the user to see what note is being played when they change the frequency.



This Diagram represents how the switches are connected and how a note is stored on a flip-flop corresponding to that switch.



This diagram portrays how the center button is connected to The counter which is ultimately connected to the seven-segment Display.

REFERENCES

[1] Llamocca, Daniel. "DIGITAL SYSTEM DESIGN VHDL Coding for FPGAs Unit 7." *RECRLab*, Electrical and Computer Engineering Department, Oakland University, www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html.

[2] "Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics." *Xilinx.com*, Xilinx, Inc., 13 Apr. 2017, www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf

[3] "VHDL Code for Clock Divider (Frequency Divider)." All About FPGA, 10 Jan. 2018, https://allaboutfpga.com/vhdl-code-for-clock-divider/