Monophonic Octave Keyboard

Audyn Curless and Joon Cho

August 26, 2014

Project Description

The project aims to create a one octave keyboard, where each note is mapped to a specific button. The keyboard is activated and deactivated by an on/off switch. Whenever a button is pressed, the note is displayed on a 7 segment display, and it is also played through one earphone.

The project was lead by Audyn Curless and Joon Cho for the class Digital Electronics (ENGS 31).

The report goes over the design and use of the project, which utilizes all of the features listed above.

Table of Contents

1. Introduction

Problem

The goal of the project was to create a digital multi-octave keyboard that can create sounds of notes through circuit logic. Another goal was to display those notes on a 7 segment display and utilize and on/off switch.

2. Design Solution

2.1 Specifications

Functional Description:

The keyboard will play every note of an octave (12 notes in total) and the octaves available are C1, C2, and C3. Every button will be mapped to a note, and the note will be played as long as the button is held down. Another button will be dedicated to switching between the 3 octaves Sounds are outputted through one earphone attached to the FPGA through a PMODAMP. Furthermore, the keyboard can be turned on and off with a switch. When off, no sounds will play no matter what button is pressed. When on, all functionality resumes. The note and octave being played are displayed on a 4, 7-segment displays and the note becomes displayed only upon a button press.

Inputs

- 12 Buttons that are each mapped to a note in the octave.
- On/Off Switch that turns the keyboard on and off.
- Octave button to switch through octaves.

Outputs:

- 4 7-Segment displays that presents the note being played and its octave.
- Earphones which plays notes upon button presses.

2. Design Solution

2.1 Specifications

Block Diagram

The black box diagram shows the inputs and outputs of the keyboard. The basic top level digram shows the datapath and control of the keyboard. The controller takes the button and octave inputs and outputs an encoding of the note value to the display. It also calculates the increment value and holds a counter that sends an enable signal to the phase accumulator at a rate of 24KHZ. Another signal, reset_sine_count is sent whenever no buttons are pressed. The phase accumulator either sets the increment and outputs 0 upon a reset signal, or increments with this value and outputs a value to the Sine LUT upon an enable signal. The output becomes the index to access a value in the Sine LUT. The samples from the LUT are then outputted to the PWM, which produces square wave pulses at a desired frequency.The PMODAmp takes that PWM output and creates sound through the earphones.

2. Design Solution

2.2 Operating Instructions

Supplies:

- Xilinx ISE (program)
- Diligent Adept (program)
- 1x Digilent NEXYS 3 Spartan 6 FPGA
- 1x Digilent PmodAMP2
- 3x Button Module

Set Up:

1. Open the keyboard project in ISE and generate the programming files. Watch for errors.
2. Insert the Digilent Button Modules into JA1, JB1, and JC1 ports of the FPGA.
3. Insert the Diligent PMODAmp2 into the JD1 port.
4. Connect the earphones into the PMODAmp2.
5. Open Adept, and select the bit file generated in step 1 to program the FPGA.

Use:

- To turn On/Off
  - Turn the switch (V16) up to turn on. Sound will be played when on. Turn the switch down to turn off.
- To play notes
  - Press any button on the attached button modules to play a note. Only one note can be played at a time and the note will play only as long as a button is pressed down. The buttons are placed so that notes are mapped in ascending order, where the left-bottom button is A, then the left-top button is B_flat, then the right-bottom button is B, and then the right-top button is C.
- To change Octave
  - Press the button (b8) to cycle between the three octaves. The octaves cycle in this order: 3,2,1,2,1 and so on.

3. Evaluation of Design

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.

4. Conclusions

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.

4. Acknowledgements

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.

6. References

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.

7. Appendix

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.

7. Appendix

Analysis

The solution was simple, and so there could have been a lot of improvements.

The first improvement we could have made is to add more notes and more octaves. A small sample of possible notes were utilizes and the FPGA has space for a lot more.

Using buttons was not the most practical nor elegant solution for a monophonic keyboard. The buttons were awkward and it would be hard for anyone musically inclined to utilize the device.

Conclusion

The goal of the project was to make a multiple octave keyboard to play notes. The onoff switch would enable and entrance and exit that would notify the user. The notes and octave being played would be displayed on the 8 segment display.

We were able to accomplish all that, but did not add anything extra.

People pursuing this project in the future should make sure to tackle functionality one at a time, instead of holistically as we did. First test that the buttons register and that the program creates a useful signal indicating which button has been pressed. Then build the phase generator and make sure that it is incrementing through correctly. Then play with the incrementation, and so on.

The biggest problems we had came from asynchronous inputs. We clocked many processes that shouldn't have been clocked and clodidnt clock ones that needed to be. This caused a lot of variability and other problems.