Square Wave Generator Peripheral and its Use in Generating Music

Vincent Chow

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
November 24, 2020

Introduction

This document contains information about the square wave generator peripheral's functionality and the design decisions with an intended audience of someone with the design who wishes to extend it. The design is a peripheral that can produce sound at different frequencies and includes an application with four features to demonstrate the capabilities of the peripheral. The features include outputting a single frequency and displaying the frequency on the Hex LEDs, playing pre-programmed song, and live song recording and playback, which the user can control using switches. These are included as these are the fundamental features the peripheral should be able to do. The device meets all of the required project specification as well as the features brainstormed.

Device Functionality

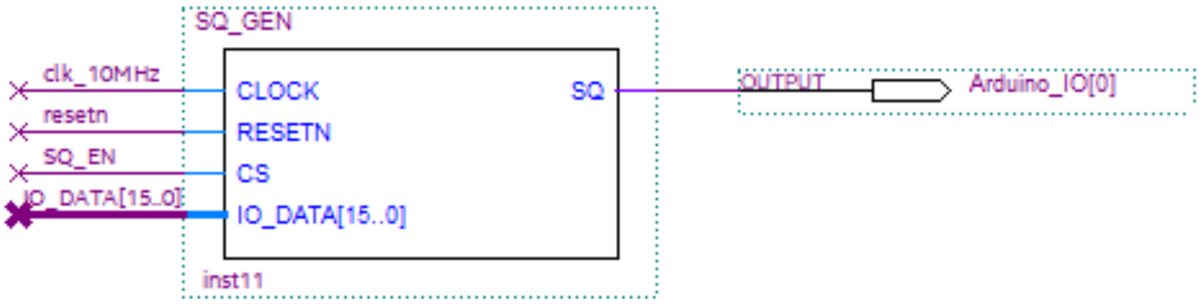
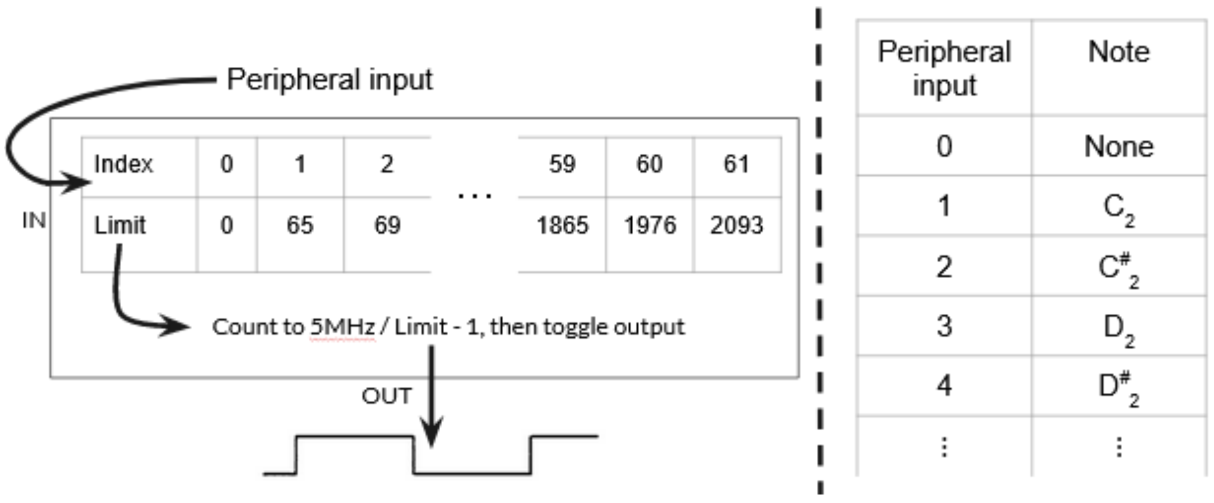


Figure 1. Block diagram of the square wave generator peripheral.

The peripheral's main function is to take a 6-bit binary input and output a 50% duty cycle square wave with a frequency corresponding to the selected note. The 6-bit binary input corresponds to the following notes: for input 0, there will be no output, for input 1, the output will be C_2, for input 2, the output will be $C^{\#}_2$, and so on until 61, which corresponds to C_7. For use with SCOMP, the user should write assembly code to input the 6-bit binary number into the accumulator, then run an OUT instruction to address of the peripheral, 0x020.



Index	0	1	2	…	59	60	61
Limit	0	65	69		1865	1976	2093

Peripheral input	Note
0	None
1	C_2
2	$C^{\#}_2$
3	D_2
4	$D^{\#}_2$
⋮	⋮

Figure 2. A high-level representation of how the peripheral converts the input to the output (left) and a mapping of which input corresponds to which note (right).

The main special feature of this peripheral is that the output notes are mapped directly to 64 numbers. This makes the peripheral user-friendly for musical applications since the user only needs to input the number that corresponds to the note they want to play.

The user should be wary of a situation that arises when a number greater than six bits is inputted into the device. In this case, the device will only read in the first six bits of the number.

The peripheral's main use in the demo was acting as an interface between the program and the audio output via the speaker through the IO address.

Design Decisions

Instead of creating a peripheral that could output every integer frequency in a certain range, we simplified the peripheral to only output frequencies corresponding to notes from C_2 to C_7 because the brainstormed application for the peripheral was musical in nature. This simplification allowed for more notes to be played versus if the peripheral were to play all the integer frequencies in a certain range, and reduced the inputs. To implement this feature we decided to use an array to store and map it with the possible frequencies for efficient access.

Peripheral and basic functions that were the building blocks were prioritized, which made extending to complicated features easier. Future work may include modifying the assembly code so that it can store more notes, or use the LEDs while playing songs to enhance the visual effect and the atmosphere.

Conclusions

Thing I would do differently for the design is to include illegal warnings, so that the user knows whether switch pattern inputted is illegal to enhance the user experience. The strengths of the design are that it implements all the basic requirements and features efficiently. A weakness of the design is that the features only include the basic functionalities and don't reflect creativity. The design process can also be optimized by dividing the work better to complete the design earlier to have more time to implement the extra features. We also should have brainstormed along the way instead of sticking with the idea we came up with originally to come up with more features.