Coord Converter Project Summary

Lizhe Zhang

Submitted
July 26th, 2022

Introduction

This document will summarize the purpose and functionality of Coord Converter (a SCOMP peripheral) that is used to convert a 2D coordinate to the corresponding NeoPixel address. We created a snake game to demonstrate that it can correctly make the conversion and its extensibility for users to make their own application. The final version of our device meets the function in our proposal, but the demo snake game has certain issue needs correction.

Figure 1. Peripheral Work Flow

Device Functionality

The Coordinate Converter peripheral receives one 16-bit data of 2D coordinate inputs from SCOMP. We use IO decoder to OUT from SCOMP to the peripheral. The peripheral converts the 2D x-y coordinate to a 16-bit NeoPixel address. The converted address is sent to a NeoPixel Controller peripheral to communicate with the corresponding pixels in the NeoPixel array. The input signal detect bit [0-4] as X coordinate and [5-7] as Y coordinate. Because the Neo Pixel has a zigzag pattern, we take different arithmetic for even and odd rows, and we use the tenth bit of input signal to indicate which arithmetic should be used. For example, if the user would like to convert (1,2), the signal he or she should input is “0000001000100010”. We recommend to use switches as the input so that SW[1], [5], and [9] should go up and others should go down to indicate coordinate (1,2).

Design Decisions and Implementation

In designing the peripheral, one major decision we made is to limit the coordinate converter's input to one 16-bit binary string. Since the NeoPixel array's size is given (32 by 6 pixels), we figured that such a small array requires only 8 bits of binary representation. We realized it's beneficial to dynamically allocate the 16-bit SCOMP registers for a flexible implementation that keeps the essential conversion functionality while minimizing the memory space and commands required for using our peripheral. Hence, we provided only one input (CC_EN) for users to conveniently inputting both x-y coordinates in one string.

Another decision is that we use the tenth bit to indicate the odd row because we encountered multi-assignment issue. If we do not take that bit to account for different situation, our peripheral sometimes give two different addresses (two pixel could light up at the same time even though we only make single assignment). With this decision, our users need to consider one more bit while writing their own application.

For the snake game, our original goal is first to set a target on the NeoPixel screen, then move the snake (a single pixel from the origin coordinate) to the target and change the color. Here, we provided a switch-control feature that allows one specified pixel to move between rows and columns. However, there are bugs that still need to be fixed that, when we try to move the snake, the target pixel might be off. Also, the snake moves faster while moving to the right than to the left.

Conclusions

The final version of our device meets the function in our proposal that convert a 2-D coordinate input to its corresponding NeoPixel address. For the demo game, our group provided a switch-control feature that allows one specified pixel to move between rows and columns so that we can move the snake to hit the target. However, it has certain bugs that still needs to be solved. My recommendation for potential users is to pay attention to their application, as our coordinate converter peripheral works

correctly, and they just need to make sure to use the tenth bit single to indicate whether the address is in odd or even row. Next time we will spend more time on debugging the demo game to fully optimize our application's functionality and distribute the group work in a balance way.