Final Project

Ly Nguyen

March 17, 2022

Programmable Devices

Introduction

For my final project, I implemented a digital clock using the altera board provided, and its 6 seven segment displays. The digital clock displays hours, minutes, and seconds in the format HH:MM:SS where it displays the time desired. There is a function to stop the clock and edit the hours and minutes like that of a watch. This stop and edit time state is toggleable, but there is no use of a state machine in this lab. Time is told in a 24-hour format starting from 0 to 23, unlike the traditional analog clock from 1 to 12.

Theory

A one-digit-counter counts from 0 to n and repeats starting at 0 again. n can be set in the code.

A clock-divider lowers the frequency of a clock to make a display, like the seven-segment display, readable between clock changes. This divides the internal clock signal on the board from 50mHz to 1Hz, or one cock cycle a second.

A seven-segment decoder takes an input and decodes it into the segments of the seven segment display.

A clock tells the desired time set. It updates every second and can be stopped to set a different time if desired.

Equipment and Method

Programs:

Quartus Prime Version 20.1.1 Build 720 11/11/2020 SJ Elite Edition

DE1-SoC Cyclone V Board

Overview:

- One digit counters used to implement each segment of the clock (6 counters for HH:MM:SS)
- Each counter depends on the output of the last for an if statement to check when it can increment
- Edge case statements were required for the hours since they did not run the same 59 display loop that minutes and seconds run
- Switch input to toggle pause/resume time and to enable button presses for changing time.

Procedure:

To start, I figured that the use components would be the best way of implementing the clock, since we previously have implemented the one digit counter through components and packages (lab 5). Already having the components of clock divider, counter, and seven segment display from previous labs, all there was to do was implement it in a way that correctly displayed and timed hours, minutes, and seconds. To start, I started out with the digit furthest to the right (HH:MM:S**S**). With this, it is in-sync with the clock divider output and cycles through 0 to 9. Next was the 2nd digit which displays the tenths place of seconds (HH:MM:**S**S). This digit only has to display from 0 to 5, and also should only increment in sync with the clock cycle, and only when the other seconds digit ticks over from 9 to 0. To do this, Another counter file was made, but with an input that took in the output of the one's place digit. Every digit in the clock had a separate counter file made for it that depended on every digit output from before it, so HH:**M**M:SS depends on HH:M**M**:**SS** outputs to check for a current 9, 5, and 9 in order to tick up. It is different for the hours though, since the first digit has to check with the second digit and vice versa. See Appendix A for the input to output process for the counters.

For hours, the first and second digits have to check each other's values so that the first digit does not go above 3 when the second digit is 2, and that the second digit can only go up to 2 then reset to 0. When implementing start and stop time to enable the time to be edited, because there are only 4 push buttons on the board, only the hours and minutes will be changeable – like that of an analog watch. The toggle was implemented with a switch on the board and an if statement to check if it was a 1 or a 0. With the push buttons being active low, while they are pressed down, they increment based on a clock cycle, the same clock input that is used to count up (so they increment every second). Again, the edge case for the hours comes up here, so if statements are required to check if the first digit is above 3, then the second one cannot increment above 1, and if the second digit is above 1, then the first digit cannot increment above 3. This edge case is to prevent an inaccurate/impossible time from being displayed.

Results

The clock works as intended! There was the idea to implement more features like an alarm system, or countdown timer system, but ultimately that was too ambitious for the amount of time spent on the project already. A small error that appeared when developing the clock was having each segment increment correctly at the correct time. Initially, I only had a check for if the digit previous to the current digit was at its max value. This however had it so the digit would increment twice, as it incremented once when the digit first became the max value, and incremented again after it would reset. To fix this, I added an AND check to see if the clock cycle was a rising edge, so then it would only increment once. This resolved the issue.

Discussion and Conclusion

Initially when planning out the clock, I had the idea of using two seven segment displays at once, so then the main entity port maps would not be so lengthy. This would allow me to use one counter for two displays, however when trying to implement two seven segment displays in one vhd file, there were a lot of errors, and they ultimately could not be solved after taking some time to debug. So the idea was scrapped, and I went to use the components from lab 5 but in a different way involving multiple different counters.

References

Lecture slides and lab manual

Appendices

Appendix A: RTL Viewer of the finalClock.vhd (main entity file)