Final Lab: FPGA Alarm Clock

Ethan Cook, ELEC 4200 - section 001

April 28th, 2024

1 Introduction

The purpose of this lab is to come up with and design a complex project that can be implemented on an FPGA. The project chosen resembles a simple alarm clock with two features: a programmable digital clock and a programmable alarm system. All code in this report is written with Verilog and implemented with Vivado 2018.1 and a Nexys A7 board.

2 Set Requirements

Before developing code for the hardware, defined requirements are needed to provide support in the development process. First, the alarm clock's interface should be easily understood by an inexperienced user. Next, the alarm clock's features should be marketable to a wide range of users. Lastly, the alarm clock should be functionally correct. For example, the clock should increment when the alarm is being programmed. Due to these requirements, the alarm clock should be simple. After requirements were made, the overall structure can be created. The alarm clock must have an interface for programming the clock and alarm. The alarm clock must also have a snooze feature and an alarm off switch. Finally the alarm clock must be able to display both the clock time and alarm time.

3 Functionality

Before implementation, multiple alarm clock interfaces were taken into consideration. The one described below was chosen due to the simplicity and ease of use. The alarm clock has five buttons and two switches that serve as input to the system. Two buttons are for setting the clock and alarm; two buttons are for incrementing the hour and minute; one button is for snooze. When the clock needs to be set, the user would hold down the set clock button. Once the time starts flashing, the user may press or hold down either the hour increment button or the minute increment button. Once the desired time is displayed, the user can either wait five seconds or hold down the set clock button until the time stops flashing. The set alarm button works similar. If the alarm is going off, the user can press snooze and this will stop the alarm and set it back by nine minutes after the button is pressed. One switch is used to turn the alarm on/off and the other is used to reset the system. The alarm clock uses four 7-segment displays to show the time. Only when the alarm is being set, the display will show the alarm time. The alarm clock also has two LED's. One is the P.M. indicator and the other is the alarm on indicator (Two more LED's were used when troubleshooting the project indicating the state of the system). When the clock is not being set, the clock increments every minute. The minute digits rollback to zero after fifty-nine. If the minute digits rollback, the hour will increment. The hour digits rollback to one after twelve. The P.M. indicator toggles when the hour hits twelve. When the alarm goes off, the four 7-segment displays and a red LED flashes until snooze is pressed or the alarm on switch is toggled off.

4 Implementation

Below is a list of a few functions and how they were implemented in the alarm clock project. The project contains many functions but these were the most important.

1. **State Handling** In order to handle the inputs and states of the system, an algorithmic state machine (ASM) is used. ASM's use control units to manage datapath operations. For this project, the control unit manages around ten flags which tell what the alarm clock should do. For example, if a user holds the set clock button, the control unit will send a set clock flag which would then tell the time to flash. ASM's are great for simplifying a system with many states and inputs.

2. **Button Handling** A major difficulty when designing a project with buttons is handling a button press. The alarm clock uses a 100 Hz clock rate to "listen" for inputs. Because of this if a button is held for more than 10 ms, the system will detect two button presses. To solve this issue, a counter is used. When a button is first pressed, the control unit will be told so and a 4-bit counter counts down. Only if the button is still and the counter's value is zero, will the control unit be told the button is pressed again. This logic is used for the hour and minute increment buttons and similar logic is used for the clock set, alarm set, and snooze buttons.

3. **7-Segment Display Handling** In order to drive the 7-segment display on the given board, a 2-bit clock with a frequency of 500 Hz is used. When the clock is zero, the first digit's signal is sent to the display. When the clock is one, the second digit's signal is sent to the display. This is the same for the third and fourth digits' signals. If a frequency higher than 500 Hz is used, the digits may not be displayed properly and if a frequency lower than 500 Hz is used, the user can see the digits turning on and off.

5 Conclusion

Overall, this was an exceptional lab to master creating and completing complex Verilog designs that can be found in the real world. Through many hours, I was able to complete the alarm clock that I had envisioned. I gained lots of experience troubleshooting the hardware design.