Digital Alarm Clock

(Ben Jackson, Devin Liu)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: bwjacks2@oakland.edu, daliu@oakland.edu

Abstract: Our project contains an attempt to develop a digital clock containing the functionality of a clock and an alarm. Switches and buttons are utilized to set the alarm times, and the activation the RGB LEDs will represent the alarm. The display used will be four of the eight, 7-segment display.

I. Introduction

The project's scope is designing and writing code that would reflect the use of a 24 hour digital alarm clock. This will be a running clock that also has an alarm. The alarm is represented by the use of RGB LEDs, which turn on when the alarm time is equal to that of the clock time. The alarm itself can be set quickly on the fly. The intent was to program a fully fledged operating alarm clock, which would have the ability to set a clock and an alarm.

II. Methodology

A. Design

The plan is to use four, seven segment displays to in the orientation of HH:MM to represent the clock. To represent this, we used the 10 millisecond counter that was demonstrated in class that originally counts in 10 millisecond increments that counted up to one minute. We modified this counter with two additional modulo counters, and adjusted the increment counter to count every 1 second. Thus we utilized six total counters that represent the HH:MM:SS, but we'll only display the HH:MM on the seven segment displays.

We used the switches to set our alarm, using switches J15 to R15 to resent the ones' minutes, R17 to R13 as the tens' minutes, T8 to T13 as the ones' hours, and H6 to V10 as the tens' hours. We used the right half of the segment displays, an0 to an3, to display our time. The values from counters are stored in a multiplexor which is then checked every one millisecond to update the segment displays.

Four 4-bit registers are used to store the alarm time. Upon the press of a push button (N17), the values on the switches are then stored in these four registers (See Figure 1, for the ASM model of our second state machine).

From state two, it constantly checks for when the values stored on the registers are equal to that of the clock, it would then move to state three. In state three, it activates our RGB LED's, one which produces white and one which produces a purple color. From state three it would wait for another push button to be pushed (P18), to move back to state one, which has the RGB LED's off and waiting for the alarm to be enabled again.

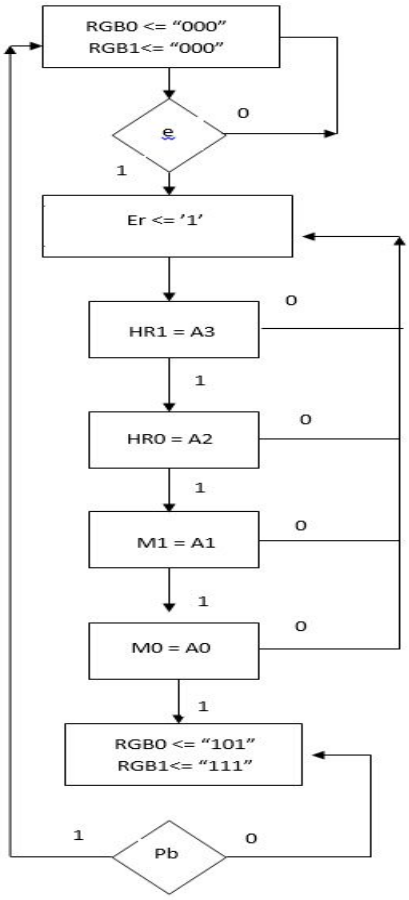

Figure 1: Alarm Control Finite State Machine

B. Setup

III. Experimental Setup

As was previously mentioned in the design section, we'll need six modulo counters; two of which that count to nine, two of which that count to five, one which counts to nine, and one which counts to two. To influence when each we'll need an increment counter that counts every one second, thus keep the seven segment displays up to date. A first, finite state machine is needed to check continuously when the time changes to update to the segment displays. We'll then use another state machine to influence when the alarm is set to the registers, and when to compare the register values to their respective time outputs to set off our RGB LEDs, The Alarm.

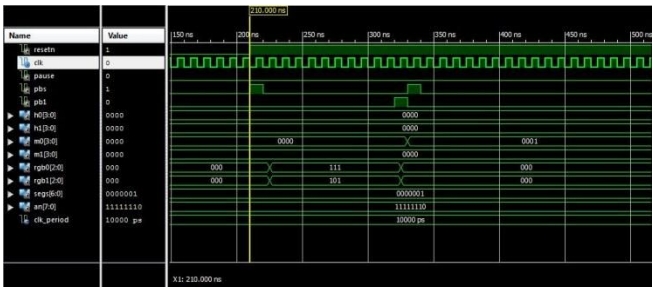

Figure 2 Timing Simulation of the alarm turning on and off, when set to 00:00

IV. Results

Originally, it was planned for this project to be a full functioning alarm clock, where we could set the clock as well as the alarm. However, due to personal conflicts, our ability to work on the project efficiently was affected, thus we settled for the clock to be more of a 20 hour timer, than an actual clock.

Also it was difficult to be able to develop a representation of a theoretical simulation (See Figure 2 for a portion of the timing simulation), as our counter doesn't begin until a second, instead of a nanosecond or milliseconds, that the test bench runs on. Added to that, it would've taken a very long for our timing simulation to run, to be able represent our clock. To simulate it completely, it would have taken over 24 hours to complete the process of the simulation. Also, due to only having one actual output, but displaying 4 values, we found it difficult to develop a way to simulate the actual counting clock in the simulation. Originally, it was assumed we had something incorrectly coded, but after implementing the code to the Nexys4 board, we determined that it was due to the out top file setup.

V. Conclusion

Overall, if we didn't encounter some of the conflicts we had; we feel we would've been able to put together our proposed scope of the project from the beginning. Theoretically, if we had started from the onset of group creation we would've been able to avoid some of the issues we encountered. However, we were able to work through and put together the project with the aid of Professor Llamocca.

Improvements that could be made to our project would be: implementing the full 24 hour aspect of our original proposal, adding in an element to be able to also set/adjust the clock, and an actual device that the Nexys4 could output sound to signify the alarm, like a speaker playing an alarm tune. Also to incorporate a visual flashing of the clock time when the alarm is going off would be desired. Theoretically, the 24 hour aspect could be maintained from the second finite state machine we had.

Adjusting the clock would require the ones hours and minutes to have a different counter with added lines codes to increase the count on the press of two pushbuttons outside.

Other additions include when the alarm was enabled, we could've displayed the alarm on the remaining left four segment displays, so that the user visually could verify which time they were setting the alarm to.

VI. REFERENCES

[1] Fall 2016 – ECE278: Digital Logic Design. (n.d.). Retrieved November 20, 2016, from http://www.secs.oakland.edu/~llamocca/Fall2016_ece278.html

[2] Fall 2013 – Workshop: Digital Circuit Design with VHDL. (n.d.). Retrieved November 26, 2016, from http://dllamocca.org/Fall2013_WorkshopVHDL.htm