WESTERN NEW ENGLAND UNIVERSITY | WNE

Department of Electrical and Computer Engineering

CPE 462/562 VHDL: Simulation & Synthesis

Project 3: Digital Alarm Clock Design and Implementation

Katelyn Charbonneau
Email: kc325844@wne.edu
Date: 11/20/2017

1 Objective

The objectives of the project are to design an implement a configurable digital alarm clock; be able to write VHDL code to design said clock using a wide variety of coding techniques; and implement the clock on the Altera DE-115 FPGA.

2 Design Procedure

The top level design block is shown in Figure 1.

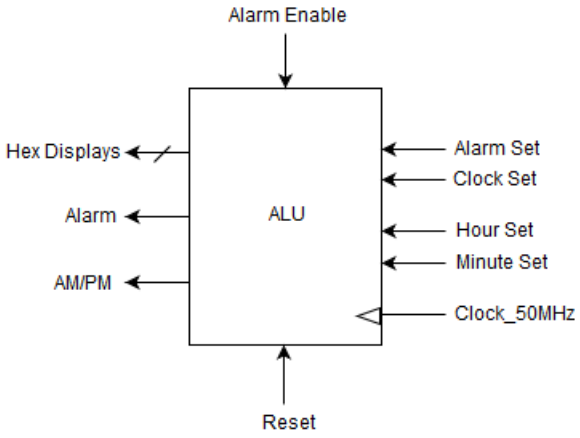

Figure 1: Alarm Clock Top Level Block

Where Alarm, Clock, Hour, and Minute Set are push buttons, Alarm Enable and Reset are switches, Alarm and AM/PM are LEDs, and the Hex Displays are 6 vectors; one for each seven segment display used in this project.

The digital clock supports setting the time and alarm time on a 12-hour clock with an asynchronous reset. When the current clock time is equal to the alarm time, the alarm will signal and can be turned off with the Alarm Enable switch.

The push buttons on the FPGA are not debounced well. As such, a custom debounce routine was written for each of the buttons; this was then included as a component in the main design. The process used in this code is shown on the next page in Figure 2.

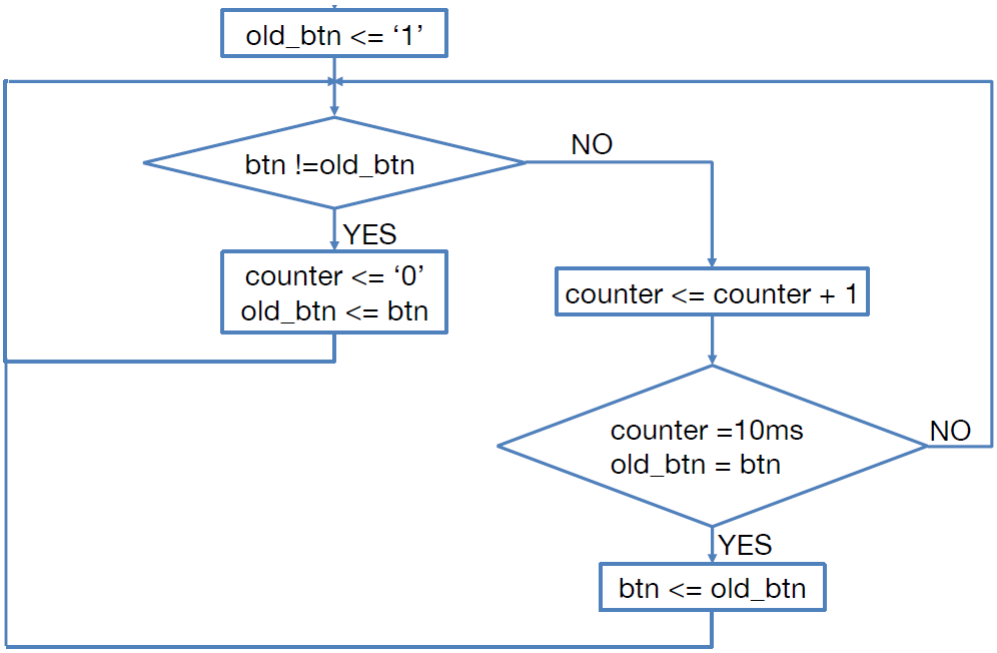

Figure 2: The debouncing flow chart

Because the clock needs to increment every second, we need a 1Hz clock to drive it. Since the FPGA's clock is 50MHz, a new 1Hz clock was made by "slowing down" the original. This was done by counting how many rising edges of the original clock are needed for a second to pass, and changing the value of the 1Hz clock when that many edges passed. The clock counts time by incrementing the one's place of the seconds value by 1 each second, which then cascades down. For example, a 9 in the one's place would next revert to 0 and increase the ten's place by 1; a 5 in the ten's place would next revert to 0 and increase the one's place of minutes, and so on. The hours reset from "12" back to "01", and the AM/PM signal changes when hours advance from "11" to "12". Hitting the reset switch sets both the clock time and alarm time to 12:00:00 midnight and locks them both at that value until reset is released.

When the clock and alarm times are equal, the alarm signal is set to high and will remain high for a minute. During this minute, it can be turned off by flipping the alarm switch, and will automatically turn itself off after a minute has passed.

3 Simulation Results

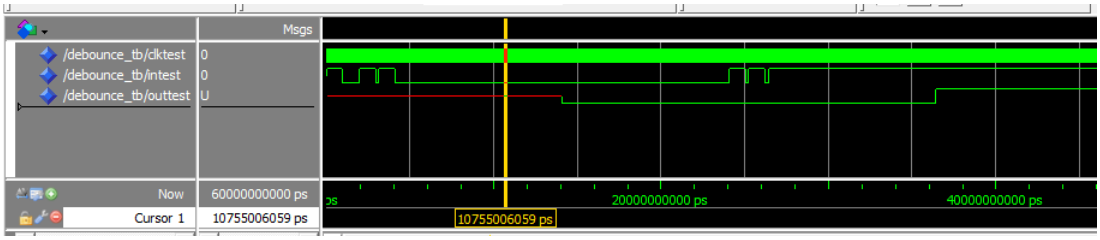

Figure 3: Showing debouncing program works with simulated bouncy input

4 Demonstration on FPGA

The design was synthesized onto the Altera Cyclone IV E DE2-115F29C7 FPGA.

- User Controls:
  - Alarm Set: Key 3
  - Clock Set: Key 2
  - Hour Set: Key 1
  - Minute Set: Key 0
  - Alarm Enable: Switch 0
  - Reset: Switch 1
- Output Displays:
  - Hours: Hex 7 and 6
  - Minutes: Hex 5 and 4
  - Seconds: Hex 3 and 2
  - Alarm: Red LED 0 (LEDR0)
  - AM/PM Indicator: Green LED 8 (LEDG8)

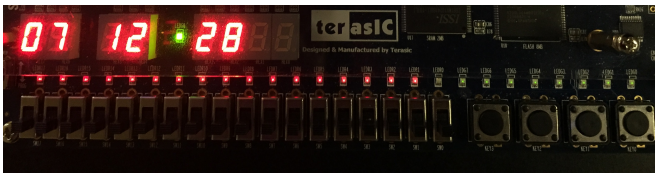

Figure 4: Displaying clock time

Figure 5: Displaying alarm time

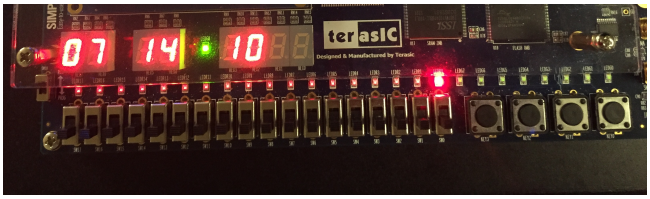

Figure 6: Alarm is set high

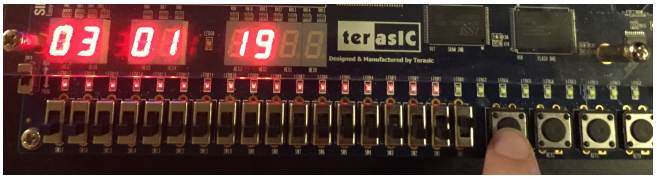

Figure 7: Alarm is set high, but the switch is set to turn it off

Figure 8: Reset is set

5 Conlusion

The project simulated and synthesized correctly. The code required sufficient knowledge of structural, concurrent, and sequential VHDL. A package was designed to compartmentalize the code to make it easier to read and debug. Recently learned functions and procedures were used within this package, as well as the declaration of the debounce component to make the main code cleaner. Some improvements need to made. One of these is the way the alarm status/enable switch functions - it could operate in a much more user-friendly manner. Also, the clock pauses while the user is changing the time (not the alarm time). Again, this could be a user-friendly improvement to make to the code to either allow the clock to continue ticking while the Clock Set key is pressed, or force the seconds to ”00” while this is happening so the clock will always be set to an even minute.