# EEL 3701 – Digital Logic and Computer Systems

# Design Problem 3

# Greg Bolling

# UF ID 64911960

## Problem Statement

The goal of the design problem was to solve a strange clock problem for the Muxians. These are a strange group of people but the seem to work a lot like the clock in Lab 04.

## Design

I took the clock from lab 4 and started to remove signals that didn’t do anything. I remove all but Switch 5, Pushbutton 1 and Pushbutton 2 (added this one in) and then had the display changed. I connected the buzzer and didn’t use the pulsing of the buzzer because it as hard to tell when it went off exactly.

Form Lab 04

The state machine uses a set of blocks in the design:

TwoDigitDisplay This module takes in two digits, in BCD format and displays them on two seven segment displays.

ClockCounter This module is responsible for generating the right next count for either the alarm or the time. If it’s the time, this has an enable signal to allow the time to increment. If it’s the alarm, it only increments using the fast increment inputs. The time has the same fast increment inputs to allow it to be set. This is used twice in the design. Once to create Alarm and once to create Time.

FSM This is the finite state machine that drives the entire clock. It has four states as described above and has a transition diagram as shown below.

SecondMux This is a mux function that takes the four seven segment inputs (that are either the time or the alarm) and converts them to digit displays that are sequenced one after another to the display using the segment selects and the seven segment values (A,B,C,D,E,F,G). It also takes in the control signals from the FSM and outputs the alarm / lights. The output is borrowed from a previous lab.

Slow\_Clock This is a process that creates a 1 Hz clock.

I then updated the FSM to add states that set and clear a register that keeps track of whether the alarm is set. To add the alarm, I figured I needed to either add a new counter or just add 10 states to the state machine (one muxian hour). It was easier to add ten states, so I added ten states and had the state machine tell the output mux what to display. It also told the output mux to hold the old clock value whenever the alarm was not sounding so when the alarm sounded it had the old value and the put that value on the display when the alarm was in one of the new ten states.

I modified the BCD to Seven Segment decoded to display the strange AM, DM and PM in the third digits. I also did this by adding a decoder to see when muxian hours were 1,2 or 3,4 or 5,6 for AM/DM/PM.

I removed the 0..59 second counter and made the minutes count with one digit ever second. Then the hours are really the minutes. Then the minutes only count 0..9 and back to zero again and the hours count 1..6 and back to 1 again. It also conditionally displays an AM/DM/PM character.

I changed the Alarm to be a state of being set or cleared and stored that condition in a register.

I changed two LED’s. LED0 is whether the alarm is set. LED1 is whether the alarm is sounding.

I connected PushButton 1 to the Hours Set and PushButton 2 to the Minutes Set. Depending on which one is pushed, the state machine either goes to SetHour or SetMinute. The counter had to be changed to make it count 1..6 in the Muxian Hours and 0..9 in Muxiam Minutes.

## Implementation

All of the designs were implemented using VHDL and similar to previous labs. The slow clock was used most of the time (labeled Clk1Hz.) This was because its different than the other clock called clock and runs at 4 MHz.

FastMinTime set the time minutes at once per second. Minutes count 0..9 only one digit. I reused the Lab04 design and only used the lower digit.

FastHourTime and set the time hours at once per second. Hours count 1..6 only one digit. I reused the Lab04 design and only used the lower digit.

CntEn is if the clock counter should count or not. It won’t count when it’s stopped.

Select Time ALn is high when the time is to be displayed, and low when the alarm is to be displayed

Buzzer directly sounds the alarm from the state machine.

SetLED0Off turns LED0 off and is for the alarm state.

SetLED0On turns LED0 on and is for the alarm state.

Led1Out is high when on, low when off and it so show that the alarm is sounding (the buzzer is going off.)

SecondMux handles the seven segment display rotation. It also has the ability to drive the Alarm in a cycle but I didn’t use that feature because it wasn’t what the design problem said to do.

I created a top level VHDL file called DesignProblem3 that was the same as Lab04 but I copied over all the VHDL code and started making changes. This block puts the other blocks together.

Two Digit Display Module was modified to respond to the AM/DM/PM symbol and produces that on a 1010, 1011, 1100 value on the display. In addition, 1100 is mapped to turn off all LED’s.

Connection

This is the same as Lab 04 and I created a diagram to show how it connects and is the same as the description above. Some signals aren’t shown because it was hard to read with too many signals.

