Will Snider
wisn3972@colorado.edu

2350 Project 2 report

Problem Statement

The problem was to create a design that counts the first 99 days of the year, displays that as a month and date, integrates leap year functionality through SW[9] and has rollover. The speed of counting is determined by KEY[1], switching between a 2hz clock when unpressed, and a 5hz clock when pressed. KEY[0] was to be used as a reset, setting the day back to 1. LEDR[0] is to be used to represent the updating of the day, following whichever clock is currently selected, blinking on at every positive edge.

Theory of operation

The project uses four modules and one parameter file (switching clock rate depending on if it is in a simulation) the top.v module takes all our outside inputs and outputs to the hex displays and LEDR[0]. Both the 5hz and 2hz clocks are instantiated with clock_divider.v, and a multiplexer is used to determine which one is used in the actual counter based on the state of KEY[0]. This clock is passed to our daycounter.v module in order to increment the day counter. Two intermediate variables are used to control the LED, with one being flipped on the posedge of the clock and one on the negedge, the two variables are XOR'd together in order to have the LED follow the posedge of the day clock. Using the BCD output of daycounter.v we can calculate the decimal value of the day and use that to determine the month as only four states need to be described. Knowing which month it is we can determine the day of the month by subtracting the total days before that month from the day counter. In order to output to the HEX displays, the ones and tens places needed to be separated, again this can be done with if/else as there are only four states for the tens place (eg. if (day <30 && day >= 20) then tens = 2; ones = day - 20;). All values are outputted to sevenseg.v, which has been seen in previous projects, in order to display them on the HEX displays.

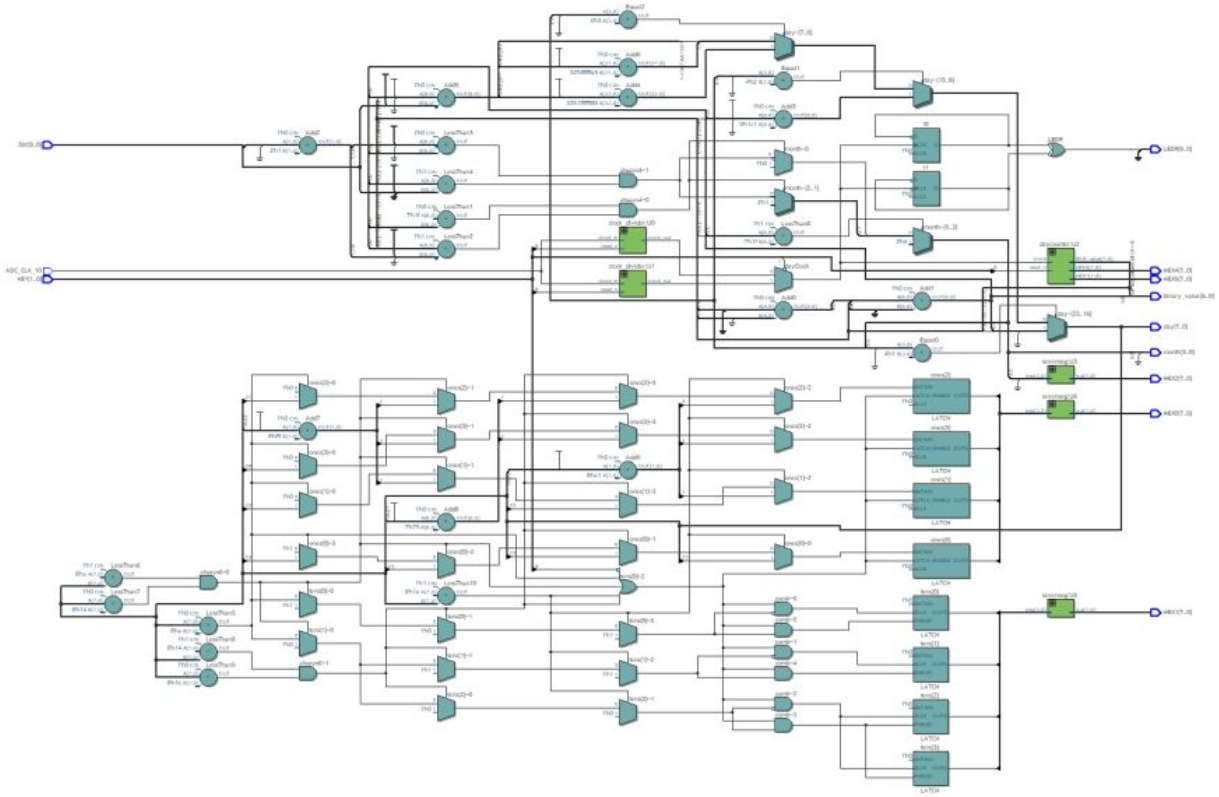
clock_divider.v works by receiving a divide by parameter, which is then used as a number to count up to, flipping the new output clock whenever this number is reached.

daycounter.v is relatively simple, taking the clock from clock_divider, BCD_value[3:0] is incremented each posedge and rolls over when the decimal value is 9, when rollover occurs, BCD_value[7:4] is incremented, both rollover at 99. A separate variable BCD_t is used for BCD_value[7:4]'s output as when the tens place is 0, the hex display needs to be blanked, and not show 0. If it is 0, BCD_t gets set to 1111, rather than 0000. A minor change was made in sevenseg.v changing the case for F(1111) to output a blank display instead in order to achieve this goal, as F was unused anyway.

Hierarchy

- top
  - clock_divider:U0
  - clock_divider:U1
  - daycounter:U2
    - sevenseg:U1
    - sevenseg:U2
  - sevenseg:U3
  - sevenseg:U5
  - sevenseg:U6

Block diagram



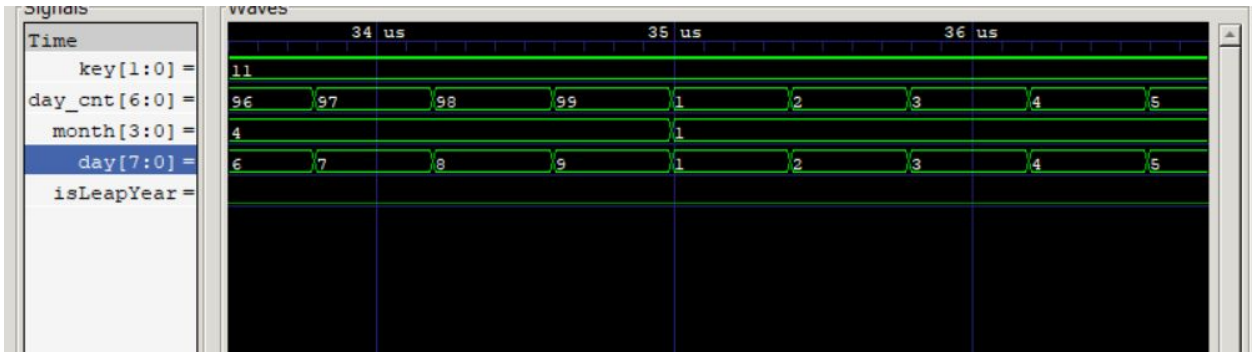
Testbench Operation

Our testbench instantiates our top module, first testing our reset at t=5, setting key = 10, the key is set to 11 at t=7. We then go into the 5hz clock mode by setting key to 01 at t = 17, and then back to the 2hz clock 3000 intervals later in order to test the frequency changing. After the clock rolls over, SW[9] is turned to high in order to test leap year functionality.

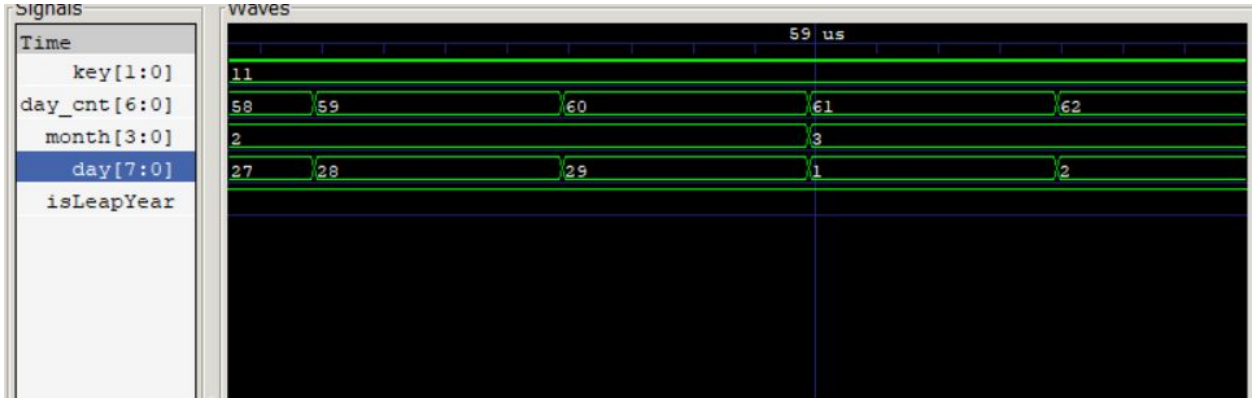
GTKwave outputs

Showing Reset:

Showing Rollover:



Showing Leap Year:



Showing Frequency Change:

Summary

All design goals were met, project functions as was required in the specifications.