EE 316 Lab #6 Report

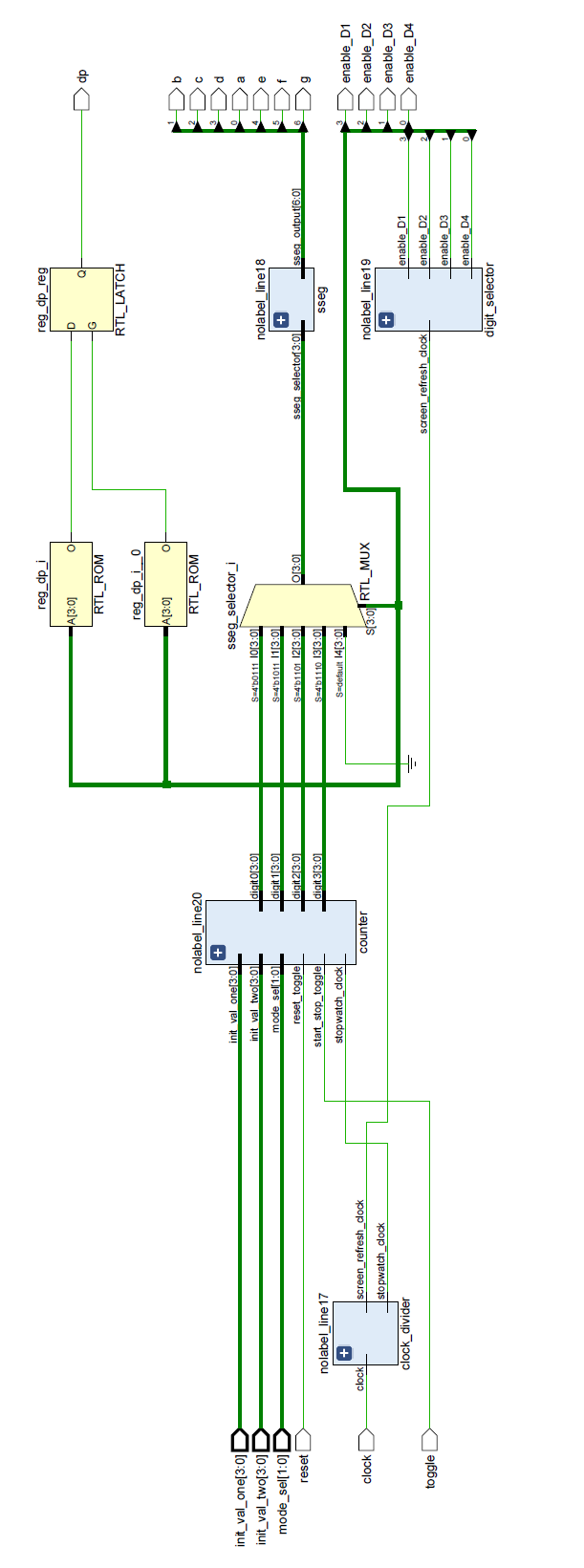
# Documentation of Design Process

I began my design process with laying out each of the pieces I’d need to complete the project. These were: a timer that could count milliseconds accurately, a counter that could use this timer to increment or decrement, a way of displaying this counter to the display, and a way of inputting the mode selection, initial values, and the start/stop and reset buttons. These already divided themselves up pretty logically into modules that could function independently of each other for the most part. Obviously, there would be connections between all the modules and building these were the next step. The slow clock would need to travel from the clock divider module to the counter module, and the digits being incremented in the counter module would need to travel to the display module. And then all of these would be conditional on the inputs received from the reset, start/stop, and mode selection buttons and switches.

My first functional attempt at this only had 3 modules, a clock divider, a display module, and a main. These came almost directly from my initial planning. However, this made keeping track of where things were, and how they were connected very difficult. I also had a lot of repetitive code with small adjustments. It worked for the most part, but debugging was a nightmare. The button debouncing wasn’t working at all and displaying the digits would only work for counting up. After messing with this iteration for a while I took a step back and decided some code cleanup would be necessary before I could continue debugging. I added an extra module and divided up the work each had a little differently. My clock dividing module now created two separate clocks, one for the stopwatch, one for the display. At first, I hadn’t even considered the need for two independent clocks each generated from the board’s clock. I thought I would only need the capability of timing the milliseconds. These could now be adjusted very easily and could not function independently of a reset button. Then I copied over the SSEG module we made in Lab 4 and made that its own module. The logic for switching between which digit to display went into its own module. Finally, the counting logic became a completely independent module, which made attacking the reset and counting down issues much easier. I had a separate main that tied all these together and had a small bit of logic for displaying the right digits and the period divider. I could have added this to the SSEG control module, but felt it fit better in the main.

Overall, the initial design wasn’t the issue. It was the fact that I rushed into it without enough proper planning, which hurt me when it came to debugging the finer points of the lab. Most of the parts we needed we had already coded in previous Labs but putting them together wasn’t as simple has copy-pasting them into a new project. The fact that there were separate modes with independent behaviors meant you had to redesign the overarching logic from the ground up.

# Processor Architecture



# Verilog Codes and Constraints

## Stopwatch Main



## Counter

**module** counter**(**

**input** stopwatch\_clock**,**

**input** start\_stop\_toggle**,**

**input** reset\_toggle**,**

**input** **[**1**:**0**]** mode\_sel**,**

**input** **[**3**:**0**]** init\_val\_one**,** init\_val\_two**,**

**output** **reg** **[**3**:**0**]** digit0**,** digit1**,** digit2**,** digit3

**);**

**reg** button\_n\_ff**;**

**reg** start\_stop**;**

**reg** reset\_n\_ff**;**

**reg** reset**;**

**always** **@** **(posedge** stopwatch\_clock**)** // look for the edge of the button. Use active low logic

**begin**

button\_n\_ff **<=** start\_stop\_toggle**;** //assign button flip flop from button

**if** **(**button\_n\_ff **&&** **!**start\_stop\_toggle**)** // if button\_n\_ff = 1 && button\_n = 0

start\_stop **<=** **~**start\_stop**;**

reset\_n\_ff **<=** reset\_toggle**;** //assign reset button flip flop from reset button

**if** **(**reset\_n\_ff **&&** **!**reset\_toggle**)** // if reset\_n\_ff = 1 && reset\_n = 0

reset **<=** 1**;** //assert reset signal

**else**

reset **<=** 0**;** //when the reset button is not negative edge, reset signal is low

**end**

**always** **@** **(posedge** stopwatch\_clock**)**

**begin**

**if** **(**start\_stop **==** 1 **&&** reset **==** 1**)**

**begin**

**if(**mode\_sel **==** 2'b00**)**

**begin**

digit0 **<=** 0**;**

digit1 **<=** 0**;**

digit2 **<=** 0**;**

digit3 **<=** 0**;**

**end**

**else** **if(**mode\_sel **==** 2'b01 **||** mode\_sel **==** 2'b11**)**

**begin**

digit0 **<=** 0**;**

digit1 **<=** 0**;**

digit2 **<=** init\_val\_one**;**

digit3 **<=** init\_val\_two**;**

**end**

**else** **if(**mode\_sel **==** 2'b10**)**

**begin**

digit0 **<=** 9**;**

digit1 **<=** 9**;**

digit2 **<=** 9**;**

digit3 **<=** 9**;**

**end**

**end**

**else** **if(**start\_stop **==** 1**)**

**begin**

digit0 **<=** digit0**;**

digit1 **<=** digit1**;**

digit2 **<=** digit2**;**

digit3 **<=** digit3**;**

**end**

**else** **if(**start\_stop **!=** 1**)**

**begin**

**if** **(**mode\_sel **<=** 2'b01**)**

**begin**

**if(**digit0 **==** 9**)**

**begin**

digit0 **<=** 0**;**

**if** **(**digit1 **==** 9**)**

**begin**

digit1 **<=** 0**;**

**if** **(**digit2 **==** 9**)**

**begin**

digit2 **<=** 0**;**

**if(**digit3 **==** 9**)**

digit3 **<=** 0**;**

**else**

digit3 **<=** digit3 **+** 1**;**

**end**

**else**

digit2 **<=** digit2 **+** 1**;**

**end**

**else**

digit1 **<=** digit1 **+** 1**;**

**end**

**else**

digit0 **<=** digit0 **+** 1**;**

**end**

**else** **if** **(**mode\_sel **>=** 2'b10**)**

**begin**

**if(**digit0 **==** 0**)**

**begin**

**if** **(**digit1 **==** 0**)**

**begin**

**if** **(**digit2 **==** 0**)**

**begin**

**if(**digit3 **==** 0**)**

**begin**

digit0 **<=** 0**;**

digit1 **<=** 0**;**

digit2 **<=** 0**;**

digit3 **<=** 0**;**

**end**

**else**

**begin**

digit3 **<=** digit3 **-** 1**;**

digit2 **<=** 9**;**

**end**

**end**

**else**

**begin**

digit2 **<=** digit2 **-** 1**;**

digit1 **<=** 9**;**

**end**

**end**

**else**

**begin**

digit1 **<=** digit1 **-** 1**;**

digit0 **<=** 9**;**

**end**

**end**

**else**

digit0 **<=** digit0 **-** 1**;**

**end**

**end**

**end**

**endmodule**

## SSEG



## Clock Divider



## Digit Selector



## Test Bench Code

## Constraints



# Simulation Waveforms

# 

