

ECE 3801: LAB 3
DUAL STOPWATCH

RYAN HOLLISTER

DECEMBER 5, 2006

ECE BOX: 38

1. Introduction.....	3
2. Pre-Lab.....	4
3. Design and Results.....	5
4. Test Bench Waveforms.....	10
5. Conclusion	11
6. Signoff:	12
7. References:.....	13
8. Appendices:.....	14

Figure 1 – Artist’s Rendition of the Stopwatch	3
Figure 2 - Functional Block Diagram	4
Figure 3 – Modulo-10 Counter	5
Figure 4 - Stop Watch Controller.....	6
Figure 5 - Modulo-6 Counter	7
Figure 6 - Stop Watch Module.....	8
Figure 7 - Final Schematic	10
Figure 8 - Controller Test Bench	10
Figure 9 - Modulo-10 Counter	11

1. Introduction

Using the internal clock of the FPGA it is relatively easy to determine elapsed time. In this laboratory assignment we will utilize this capability to build a device that capable of running two timers and displaying them accordingly on the seven segment displays present on the development board. An artist's rendition of the stopwatch is displayed in Figure 1

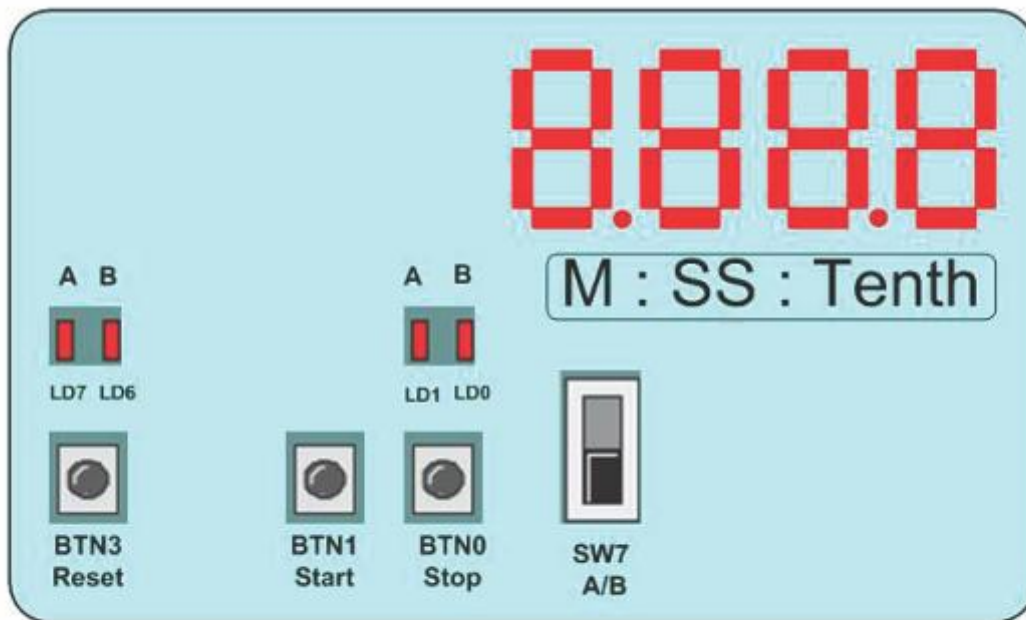


Figure 1

Figure 1 – Artist's Rendition of the Stopwatch

The figure shows the different elements that will make up the stopwatch. The switch (SW7) will select which stopwatch will be displayed on the seven segment display. The LEDs labeled LD1 and LD0 will blink if A or B is counting, respectively. The LEDs labeled LD7 and LD6 will be lit according to which timer

is active on the display. The buttons will work according to the convention of the labels in Figure 1.

The functional block diagram can be found if you refer to Figure 2. It shows the inputs and outputs of the two functional blocks of the system.

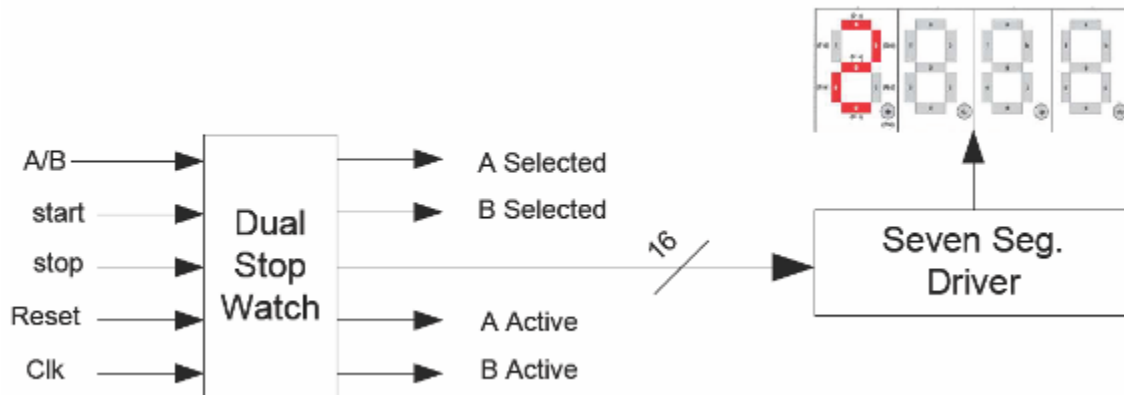


Figure 2 - Functional Block Diagram

If we were to push into the dual stopwatch module in Figure 2 we would see the functional block of dual stopwatch which consists of the controller, four separate modulo counters of the characteristic 10, 10, 6, 10 respectively from right display to left display.

2. Pre-Lab

The pre-laboratory assignment was to design the state machine for the stop watch controller and to design the modulo-10 counter. The schematic and symbol will be discussed in Section 3.

3. Design and Results

As discussed in the pre-lab I first built the modulo-10 counter prior to laboratory time. The assignment stated that it was acceptable to use Xilinx 4-bit counters in our design. As you can see in the following figure a four bit counter was used.

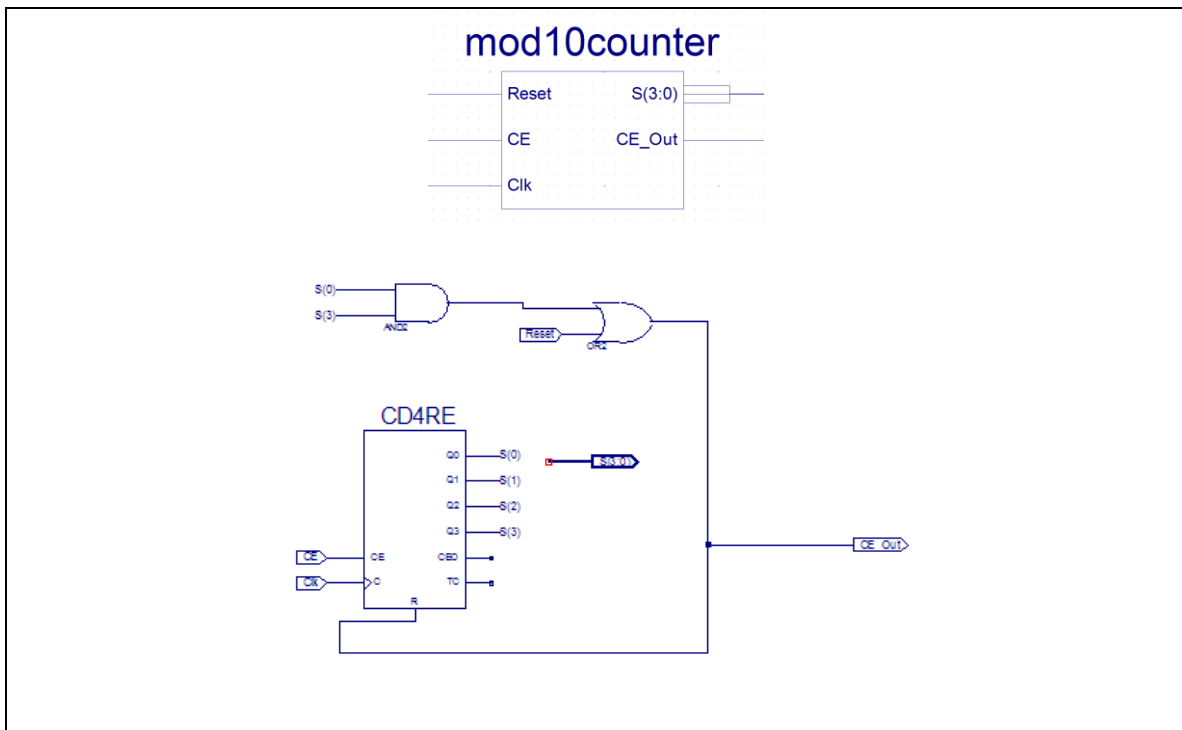


Figure 3 – Modulo-10 Counter

This design takes advantage of the carry out capability of the 4-bit counter and will assert the reset pin if bit zero and three are active or if the reset pin is active.

The next module to create was the controller for the stop watch. This was accomplished using a D-type flip-flop. There were two states that were of concern. The first was what to do when start and the stop button are pressed simultaneously. I decided that the start button would override the stop button.

This meant that I wanted the data to be active when start was pressed but there was no data and no reset. The other state was when the output is just in free running mode. Meaning that it is already outputting and stop and reset are not pressed. Evaluating these two states you can see it takes care of all the stopping and starting of the controller. The reset was not built into this module but will be discussed later. The controller module and its symbol can be found in Figure 4

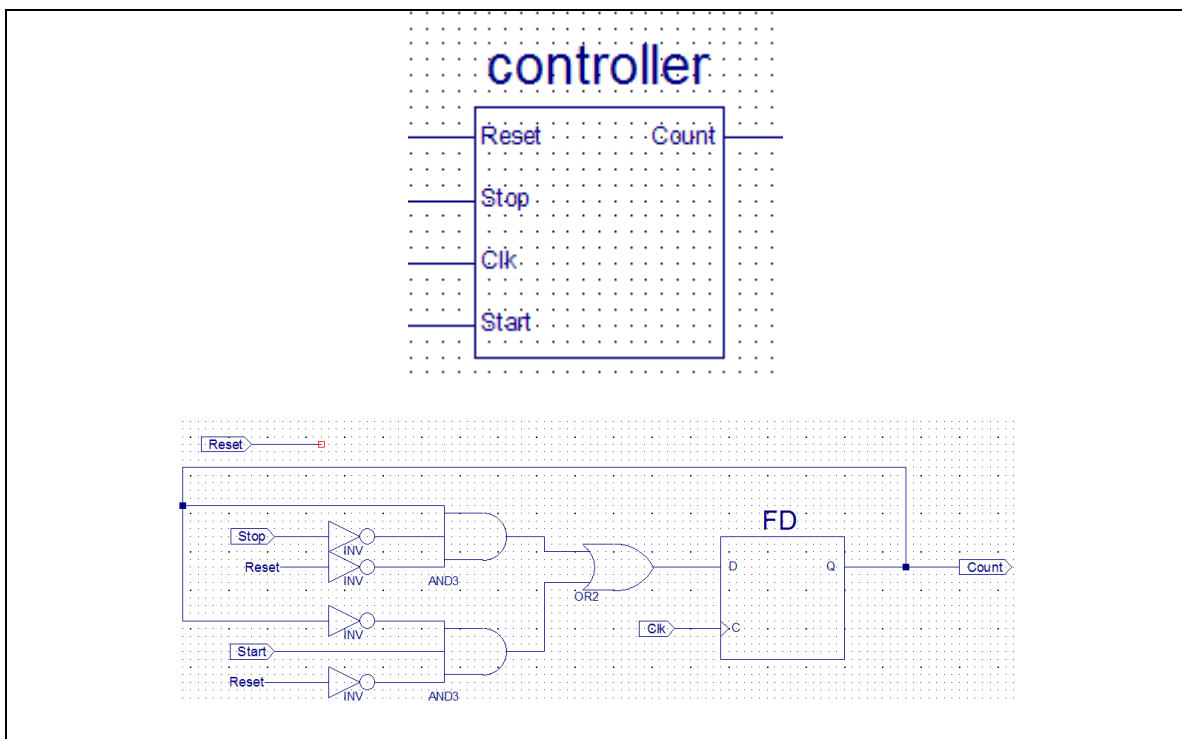


Figure 4 - Stop Watch Controller

After completing this, the assignment then called for the module-6 counter to be created. This was done very similarly to the module-10 counter only the bits to be checked were different. This needed to reset when the counter was at six (110). To accomplish this a 3-input AND gate was used to assert the reset pin. The reset portion of the design is handled within the modulo counters. The issue that I was having was that feeding the reset into the

controller was unasserting the chip enable pins on the modulo counters preventing the reset from working. There were probably other ways around this but the method I chose was to put logic on the chip enable pin of the counters to be asserted if the reset pin was pressed. This is evident in the final schematic but is also partially built into the modulo counter.

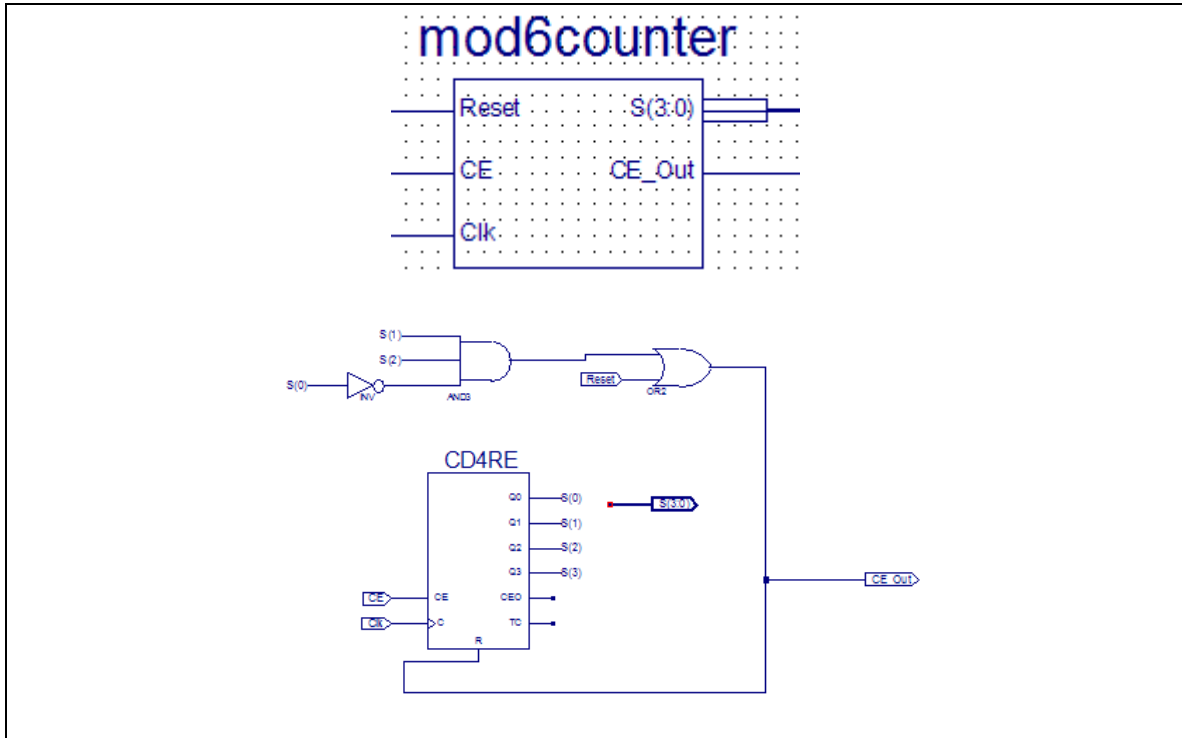


Figure 5 - Modulo-6 Counter

Once these modules were complete it was a simple task of hooking everything up. I reused the “inputmux” module from previous assignments to switch between displays. Since the buses for this part were only 4-bits wide I needed to use another one for the remaining three segments. The main stopwatch module is shown below in Figure 6. This module will be used twice to create the dual stopwatches.

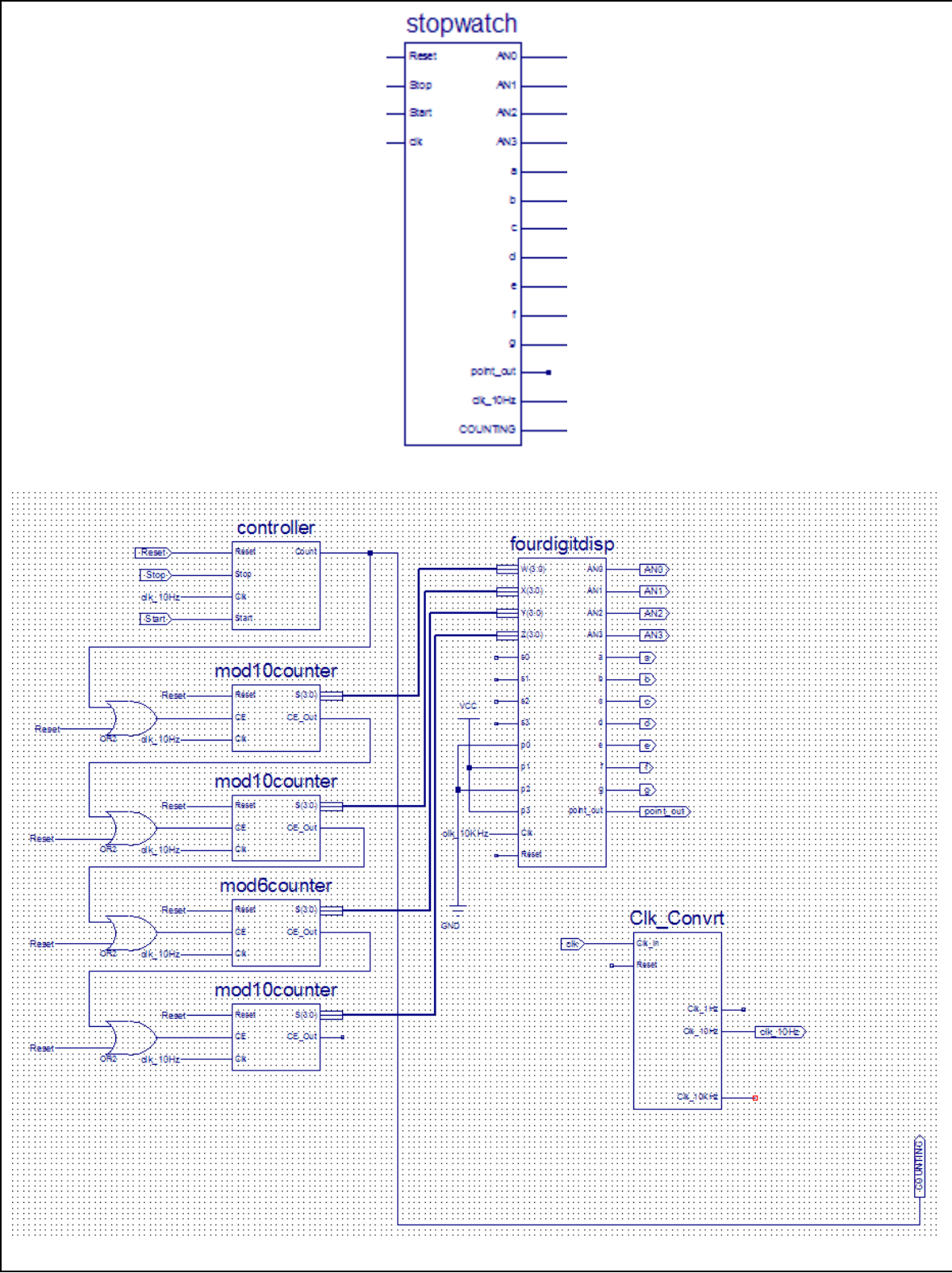


Figure 6 - Stop Watch Module

The dual stopwatch is shown in Figure 7. Beyond working with the modules we already discussed, there needed to be logic added for the LEDs and switches. Since we wanted a pair of the LEDs to blink when that stopwatch is counter I broke out the 10KHz clock and the counting signal to an AND gate. This will be high when counting is active and the clock is high. When counting is low then nothing will blink. When counting is active and the clock is low it will be off as well creating the blinking sequence. To create the state sequence it was simply a matter of taking the input from the switch and displaying its decoded value on the LEDs. LD1 was the inverse of the switch and LD0 was the actual value of the switch. All the controls needed to be filtered through AND gates to only assert them on the appropriate stopwatch that was selected. The final schematic for the dual stop watches can be found below in Figure 7.

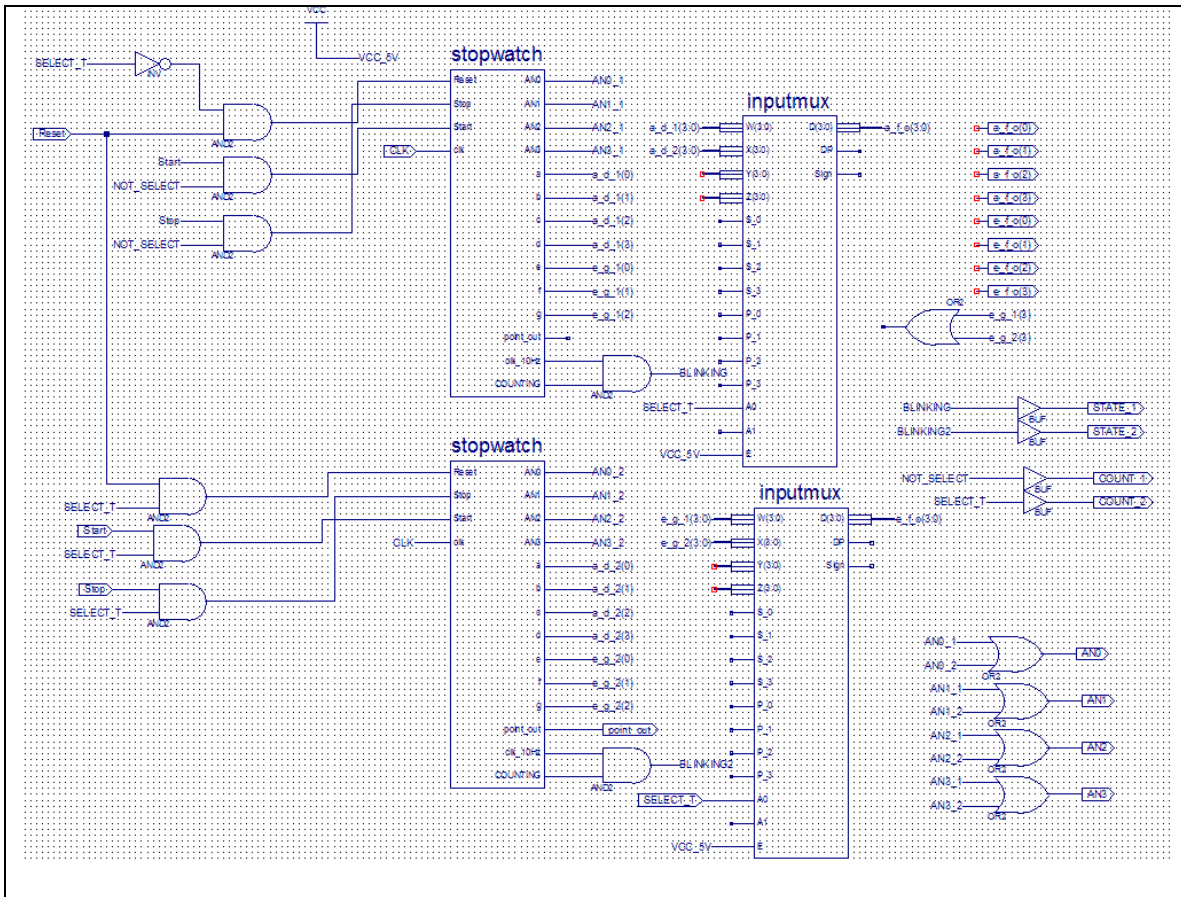


Figure 7 - Final Schematic

4. Test Bench Waveforms

Here are two examples of testbench's that were used to verify the operation of individual modules.

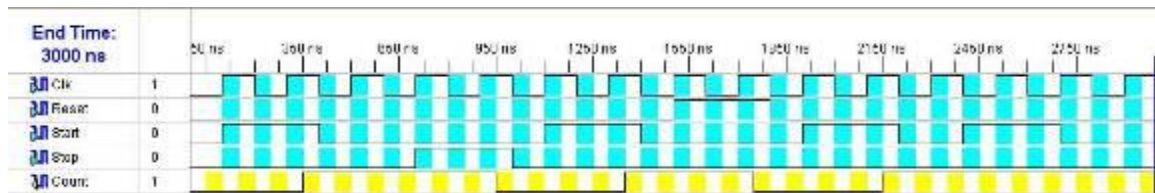


Figure 8 - Controller Test Bench

Examining the above figure one can verify proper operation of the inputs as described in Section 1.

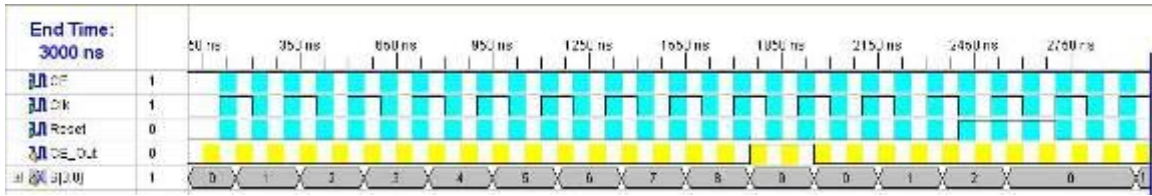


Figure 9 - Modulo-10 Counter

Examining the above figure one can verify proper operation of the inputs as described in Section 3.

5. Conclusion

I was able to complete this assignment with little difficulty. The design is rather simple once the state machine is working correctly it was a matter of knowing how the modules work and hooking them up correctly. The logic for the controls of the dual stopwatch was somewhat difficult to grasp at first. Once understood they were fairly tedious to connect but worked perfectly once complete. Overall this project was interesting and rewarding.

6. Signoff:

7. References: Some pictures in this lab were taken from the laboratory assignment:

ECE3801 – Lab3 (Dual Stop Watch)

Copyright: Feb, 2005

Written by Ahmad Hatami

Modified by Samant Kakarla

8. Appendices:

ALU Constraints File

```
#PACE: Start of Constraints generated by PACE
```

```
#PACE: Start of PACE I/O Pin Assignments
```

```
NET "a_f_o[0]" LOC = "E14" ;  
NET "a_f_o[1]" LOC = "G13" ;  
NET "a_f_o[2]" LOC = "N15" ;  
NET "a_f_o[3]" LOC = "P15" ;  
NET "AN0" LOC = "d14" ;  
NET "AN1" LOC = "g14" ;  
NET "AN2" LOC = "f14" ;  
NET "AN3" LOC = "e13" ;  
NET "CLK" LOC = "t9" ;  
NET "COUNT_1" LOC = "P11" ;  
NET "COUNT_2" LOC = "P12" ;  
NET "e_f_o[0]" LOC = "R16" ;  
NET "e_f_o[1]" LOC = "F13" ;  
NET "e_f_o[2]" LOC = "N16" ;  
NET "point_out" LOC = "p16" ;  
NET "Reset" LOC = "l14" ;  
NET "SELECT_T" LOC = "k13" ;  
NET "Start" LOC = "m14" ;  
NET "STATE_1" LOC = "P14" ;  
NET "STATE_2" LOC = "K12" ;  
NET "Stop" LOC = "m13" ;
```

```
#PACE: Start of PACE Area Constraints
```

```
#PACE: Start of PACE Prohibit Constraints
```

```
#PACE: End of Constraints generated by PACE
```