CS M152A Lab 3
Stopwatch

HUAN-SONY NGUYEN

Ryan Trihernawan (904063131)
Walter Qian (204301927)

Introduction

The goal of Lab 3 is to design a Stopwatch and implement it using the FGPA board. The stopwatch acts as basic clock, which counts minutes, and seconds, which will be displayed using the seven-segment display. The left two digits will count the minutes and the right two digits will count the seconds. In addition we will have two switches (ADJ and SEL) and two push buttons (RESET and PAUSE).

ADJ	Action
0	Stopwatch behaves normally.
1	Stopwatch paused and 'Selected' increases at 2Hz.

Table 1: ADJ switch and corresponding actions.

SEL	Selected
0	Minutes
1	Seconds

Table 2: SEL switch and corresponding actions.

The two tables represent our two switches, ADJ and SEL. When the ADJ switch is set to 1 the stopwatch halts and the selected component is increased at 2Hz to allow the user to select a new clock time. The unselected component is not increased at all and is effectively paused. The corresponding SEL switch is used to select either minutes or seconds. The RESET button will set the clock to the initial state of 00:00. The PAUSE button will pause the stopwatch when pressed and will continue when pressed again. We also need a debouncer to filter out the noise by sampling at a frequency lower than the noise.

The design of our stopwatch consists of five modules: Clock, State Controller, State Registers, Counter Controller and Display Controller. We discuss them in detail below.

Design Description

The overall circuit has four inputs. We input the two buttons for PAUSE and RESET, an array to represent the two switches and a master clock of 100MHz from which we derive the other clocks. The circuit outputs an 8-bit array for the seven-segment display to represent the time for each digit and a 4-bit array to control blinking. The first module is the clock, which takes the master clock of 100 and outputs 4 clocks (1Hz, 2Hz, 357.14Hz and 4Hz). Our second module is a state controller, which takes the switches and pause button as input to determine the current state of the circuit. It outputs a 2-bit array, which represents the four states: Normal, Paused, Adjusting Seconds and Adjusting Minutes. The third module, state register, takes the 2-bit state array and outputs four registers (minutes enabled, seconds enabled, minutes blinking enabled and seconds blinking enabled) based on the states. The first two registers determine if the two minute and second counters are enabled and the next two determine if the minute and second counters are blinking. The fourth module is a counter controller that is responsible for incrementing the minute and second counters based on the current state. The final module is a display controller that cycles through the digits using the 357.14Hz clock to illuminate the seven-segment display. A schematic of the overall design can be seen below and each of the modules will be explained in detail.

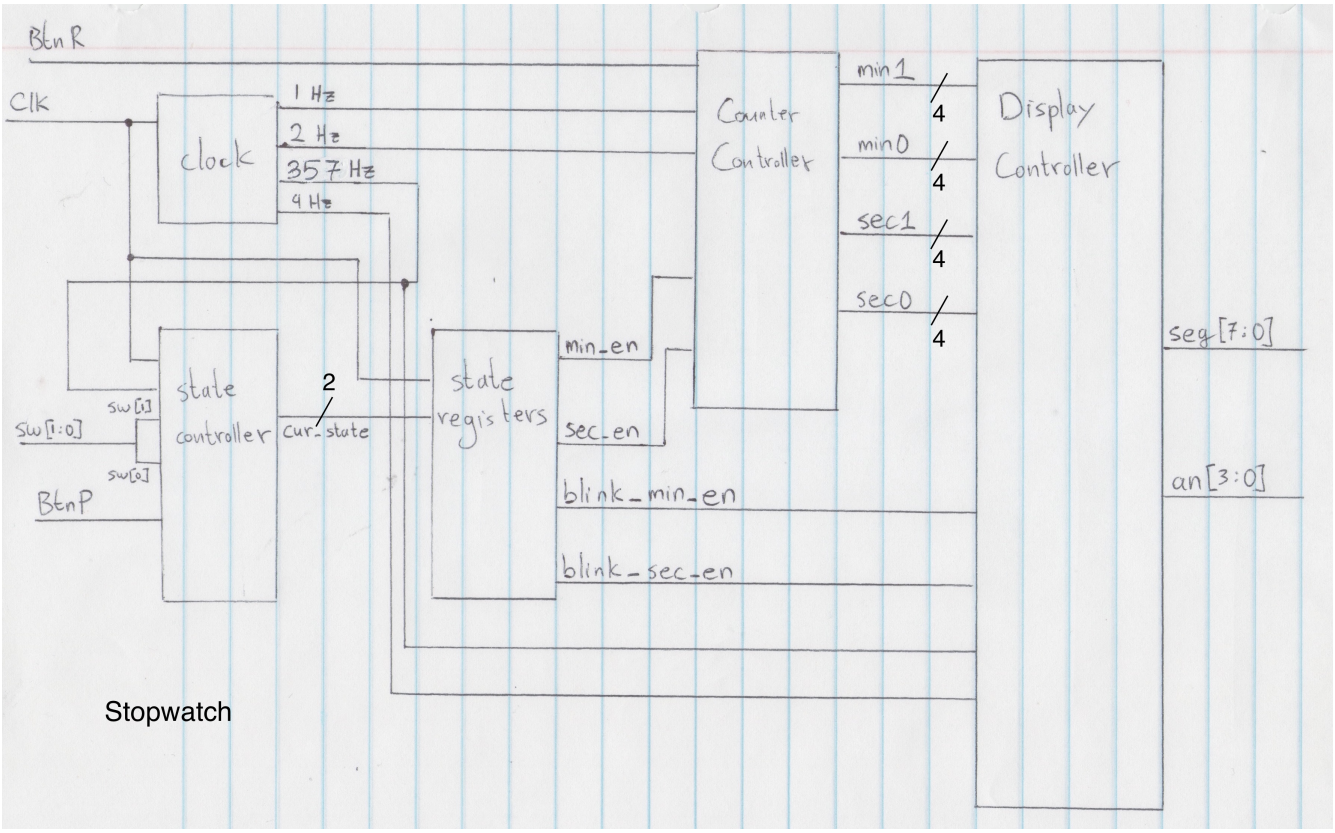

Figure 1: Circuit Schematic

The first module, Clock, takes a 100MHz master clock and from it we create four clocks with frequencies of 1Hz for regular increments of the stopwatch, 2Hz for adjusting the clock, 357.14 Hz for multiplexing the seven-segment display and 4Hz for blinking in adjust mode. To construct a clock divider we use ModNCounter. ModNCounter is a smaller module that takes the 100 MHz clock and a divisor and outputs a 1 when it resets the counter to 0. Every time the 100Mhz clock ticks up to the divisor the output clock ticks once. For example, to convert our 100MHz clock to 1Hz we divide by 50M. We do not divide by 100M because a 1Hz clock goes from 0 to 1 back to 0 in one second. To divide by 100M would lead our clock to taking one second to go from 0 to 1 and thus we need to multiply the speed by a factor of two. Similarly we divide by 25M, 12.5M and 140,000 for our other three clocks. A special mention is needed for our fast clock of 357.14Hz. Unlike the other clocks, which we assert to 1 after a period of ticks, we only use one tick because it is used as a debouncer as well. The clock module schematic is displayed below.

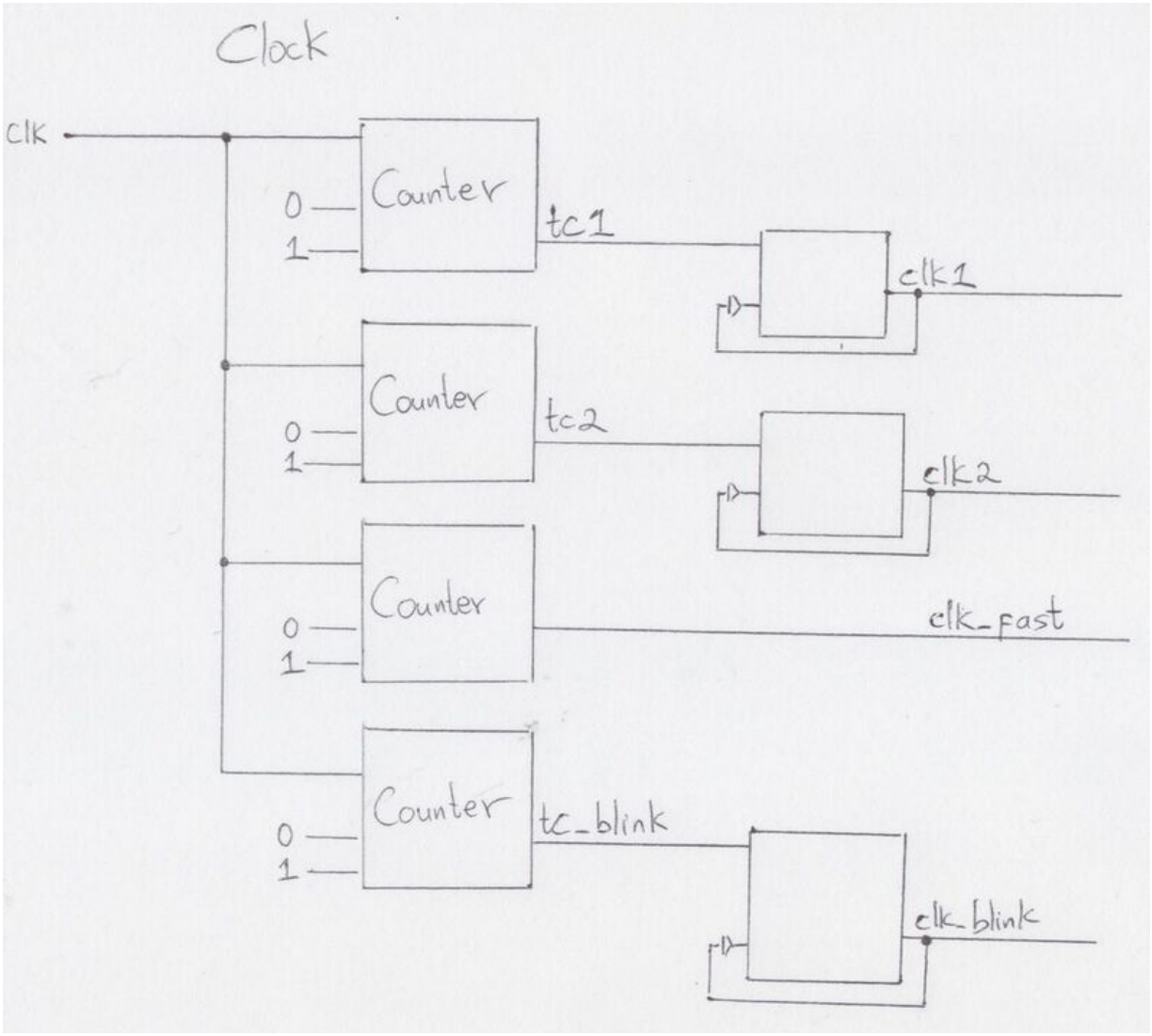

Figure 2: Clock Module

Our second module is the state controller and takes the buttons, switches and current state as input to determine the next state, which is represented as a 2-bit array. We used the fast clock as a debouncer for the pause button, similar to Lab 1 logic. We then use a MUX to get the current state from the inputs. The PAUSE state takes precedence followed by ADJ Minutes and ADJ Seconds. The default state is normal. The image can be seen below.

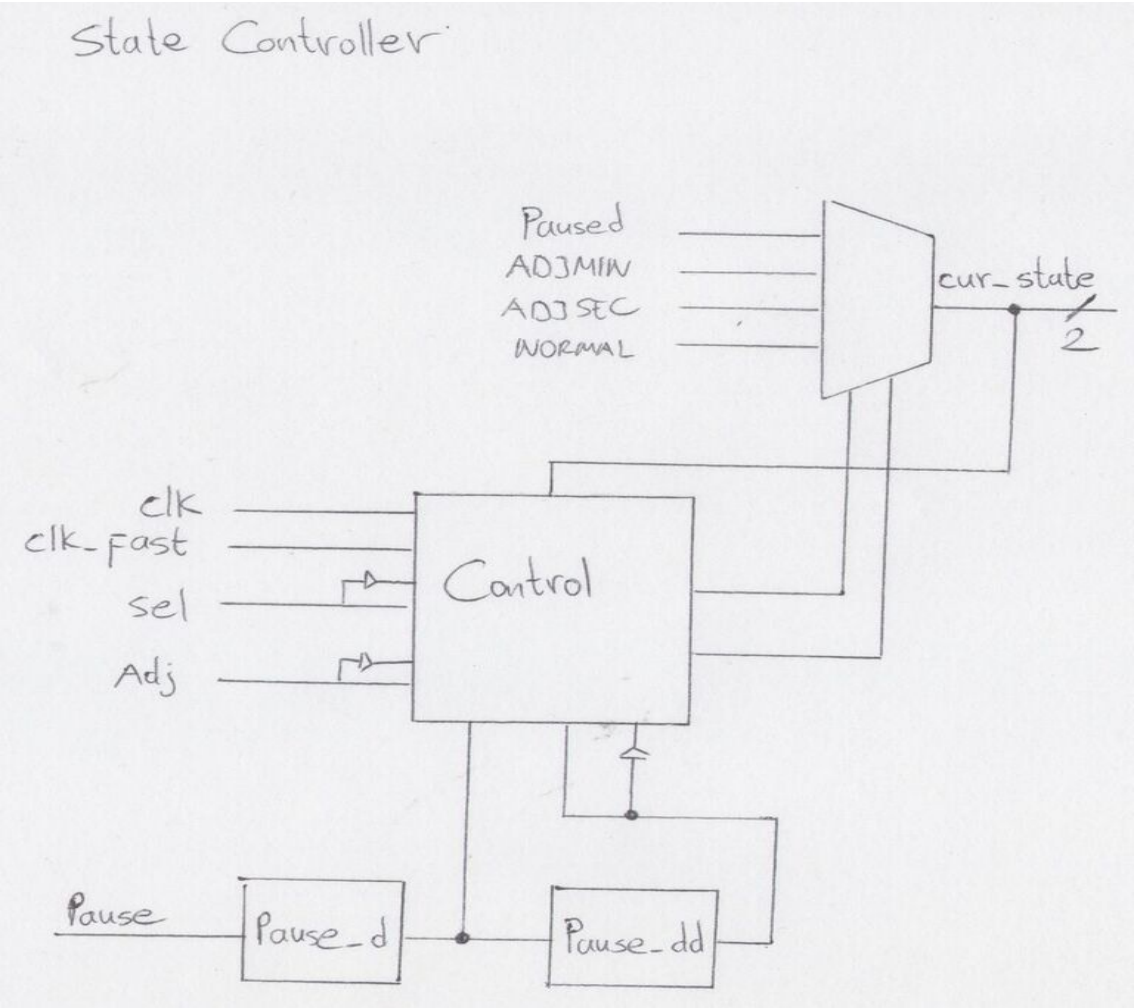

Figure 3: State Controller

The third module, state register, takes the 2-bit array current state and outputs four registers. The first two registers determine if the two minute and second counters are enabled and the next two determine if the minute and second counters are blinking. It uses a four MUXes, each to determine a register. In the Normal state the two enabled registers are set to 1 and the two blinking registers are set to 0. For the Paused state all registers are set to 0. In ADJ Minutes both minute registers are set to 1 while in ADJ Seconds both second registers are 1. Also, the corresponding blinking registers are set to 1. The logic is displayed below.

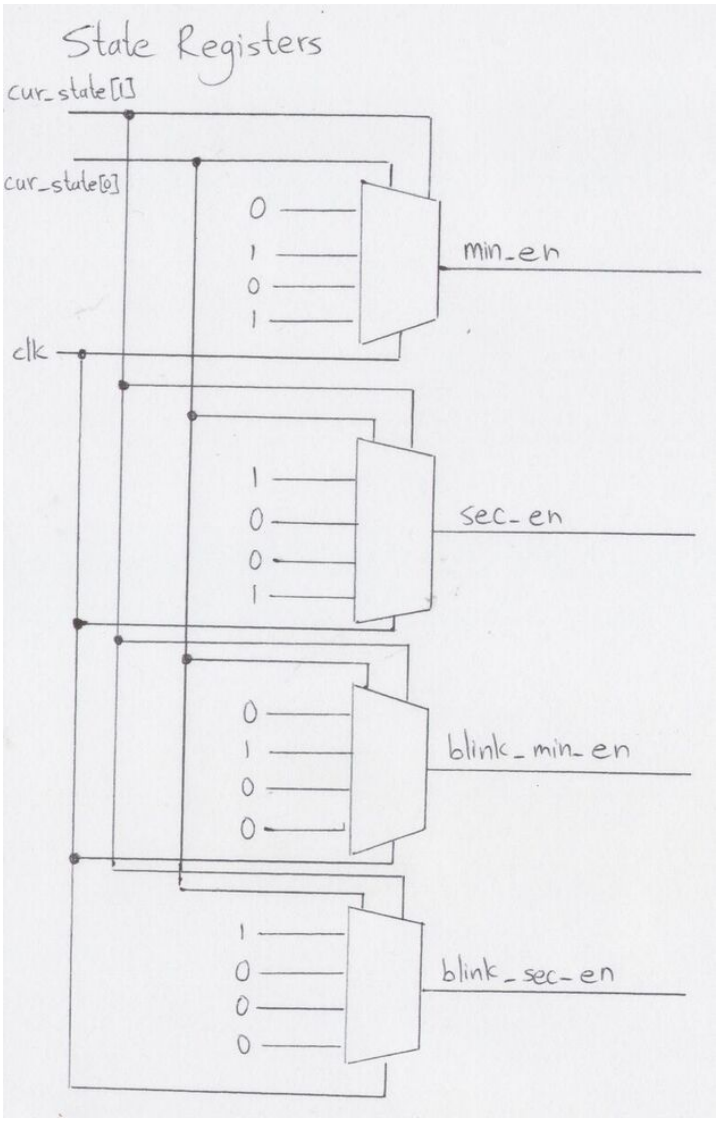

Figure 4: State Register

Counter controller is our fourth module and takes the 1Hz and 2Hz clocks along with the first two minute and second enable registers to increment the stopwatch. If both registers are set to 0 then we are in Pause state and the timer is not incremented. If only one register then we are in adjust mode for that counter and the 2Hz clock is used to increment the counter. When both registers are set to 1 the stopwatch is in normal mode and incremented normally. We use AND and OR gates to simulate the above behavior. The counter controller has four integer outputs, using a counter for each digit. We use a Mod-6 counter for the first minute and second digits. A Mod-10 counter is used for the 2nd minute and second counters. Every time a counter is reset to 0 the next digit is incremented, except when the first minute counter is reset to 0. This occurs only after 59:59 and we reset the timer to 00:00. Schematic is seen below.

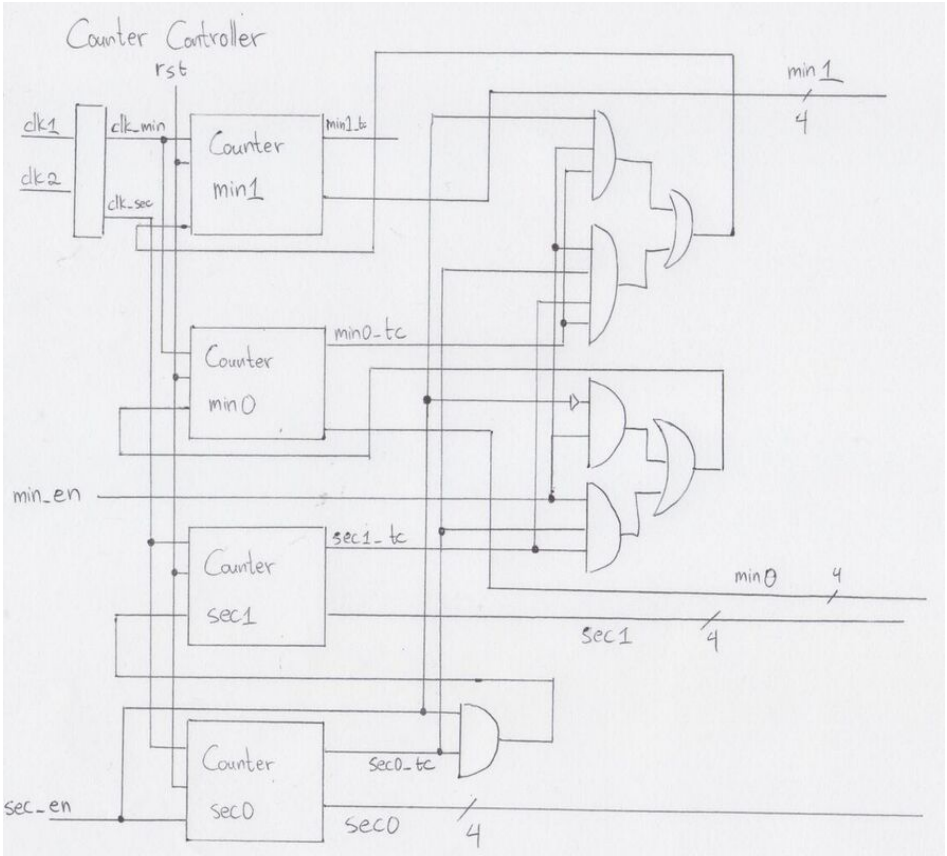

Figure 5: Counter Controller

Display Controller, our fifth and final module, is used to display the four digits. It takes the four counters in the counter controller with the fast clock, blink clock, and blinking enable registers as inputs. First each of the numerical inputs is sent into a module (DectoSeg) that converts the decimal into the segments that are illuminated in the seven-segment display. That module uses a simple switch statement for every decimal case. The fast clock (357.14 Hz) is used to multiplex the four digits. It cycles through each digit and lights it up on the seven-segment display. Since the clock cycle is faster than the human eye to us it looks like the lights are lighting up at the same time. To emulate blinking we simply skip every half a second blink (2Hz).

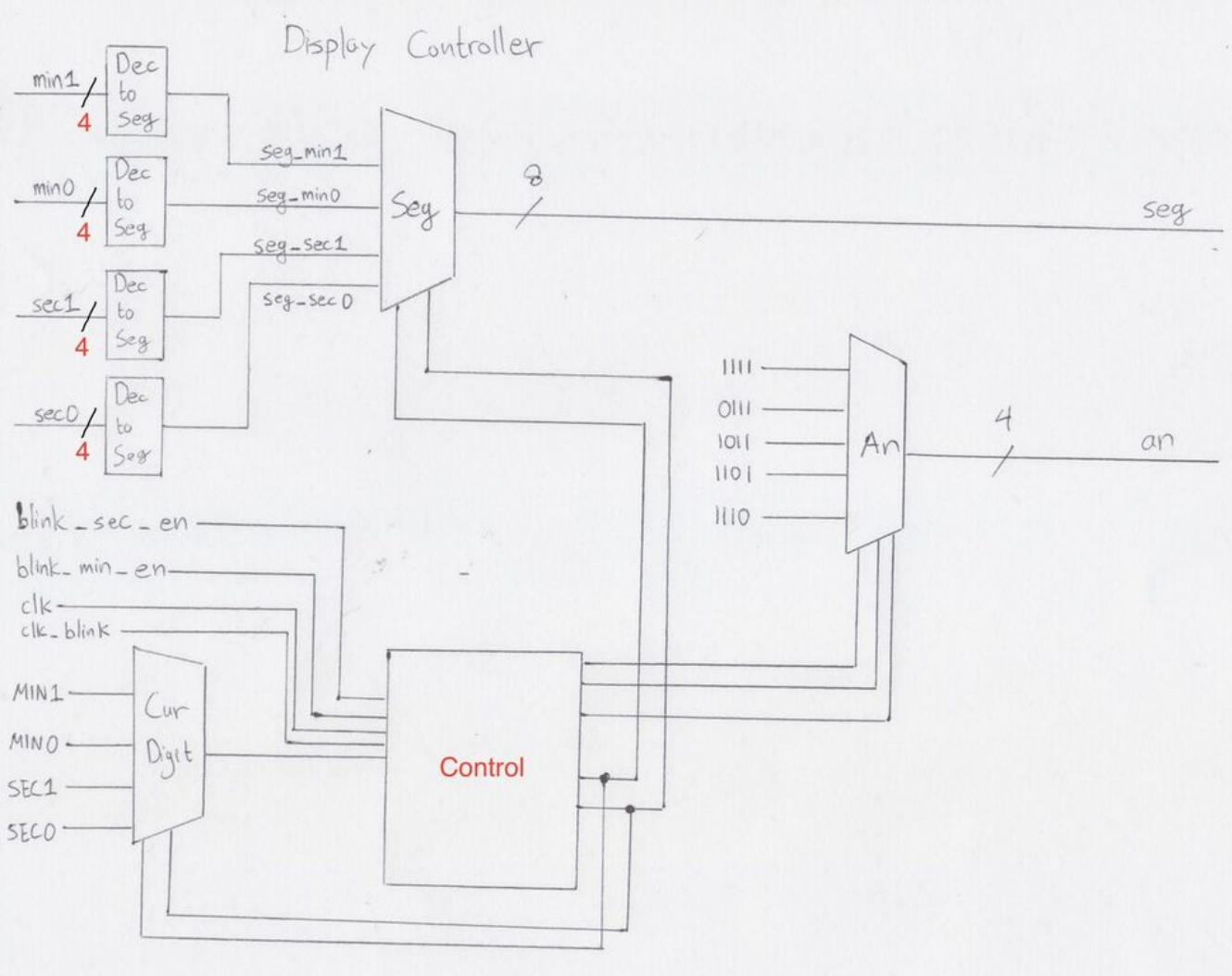

Figure 6: Display Controller

Simulation Documentation

To test our stopwatch we tested all the possible scenarios for each module. First we tested the normal state to see if the clock incremented correctly and when it 59:59 it would reset normally. We checked this by looking at the seven-segment display. The normal state worked as expected with the clock increments. Anytime a digit hit 0, the next digit was correctly set to 1. For 59:59 it was correctly reset to 0. Next we tried out the PAUSE and RESET buttons to check their functionality. The tables of our tests can be seen below. Everything worked as expected.

Clock

The clock module was tested by looking at the digits on the display and ensuring that they work as intended. This method was chosen due to the simplicity of the implementation of the clock module. The same method was used for testing the blinking and fast clocks.

State Controller

The state controller module was tested by pressing the pause button, and moving the switches. Below table shows that Pause state has the highest precedence. For test case 1 we checked that if ADJ was 0 nothing would happen even with SEL being changed. Test cases 2 and 3 checked the transition from normal to the adjust state. Test case 4 was to test switching between the two adjust states. Test case 5 was to test the precedence of pause over adjustment state. Finally, test case 6 was to test the precedence of the pause button over the switches. All the results were as expected. The debouncer also worked well as the buttons were working as intended.

Pause	ADJ	SEL	Initial State	Expected State	Result
0	0	1	Normal	Normal	Successful
0	1	1	Normal	ADJ Seconds	Successful
0	1	0	Normal	ADJ Minutes	Successful
0	1	0	ADJ Seconds	ADJ Minutes	Successful
0	1	1	Pause	Pause	Successful
1	1	0	ADJ Minutes	Pause	Successful

Table 3: State Controller Tests

State Registers

The state registers module was tested by hardcoding the inputs and observing the expected outputs as shown in the table below.

Test	Expected min_en, sec_en, blink_min_en, blink_sec_en	Result
clk = 1 cur_state = 00	1, 1, 0, 0	Successful
clk = 1 cur_state = 01	0, 0, 0, 0	Successful
clk = 1 cur_state = 10	1, 0, 1, 0	Successful
clk = 1 cur_state = 11	0, 1, 0, 1	Successful

Table 4: State Registers Tests

Counter Controller

The counter controller module was tested by observing the increments of the clock digits especially the edge cases where one digit counter resets to 0 and subsequently causes the following digit to increment by 1.

Initial Clock	Test	Expected Clock	Result
00:05	rst = 1 min_en = 1 sec_en = 1	00:00	Successful
21:33	rst = 1 min_en = 1 sec_en = 1	00:00	Successful

59:01	rst = 0 min_en = 1 sec_en = 0	00:01	Successful
01:59	rst = 0 min_en = 0 sec_en = 1	01:00	Successful
59:59	rst = 0 min_en = 1 sec_en = 1	00:00	Successful
00:59	rst = 0 min_en = 1 sec_en = 1	01:00	Successful
10:10	rst = 0 min_en = 0 sec_en = 0	10:10	Successful

Table 5: Counter Controller Tests

Display Controller

The display controller was tested by observing the seven-segment display digits such that they increment and blink as intended according to the corresponding current states.

Test	Expected	Result
clk = 1 clk_blink = 1 blink_min_en = 1 blink_sec_en = 0 min1 = 0 min0 = 0 sec1 = 0 sec0 = 0	Minute digits blink; Second digits do not blink;	Successful
clk = 1 clk_blink = 1 blink_min_en = 0 blink_sec_en = 1 min1 = 0 min0 = 0 sec1 = 0 sec0 = 0	Minute digits do not blink; Second digits blink;	Successful
clk = 1 clk_blink = 1 blink_min_en = 0 blink_sec_en = 0 min1 = 0 min0 = 0 sec1 = 0 sec0 = 0	Both minute and second digits do not blink;	Successful

Table 6: Display Controller Tests

As shown in the tables above, everything worked as intended.

Conclusion

This lab introduced us to using the FPGA board's other components including the seven-segment display. The lab spec did not specify how the seven-segment display work, thus we had to read its documentation on Digilent website. Understanding how each of the digits is displayed took some time. Moreover, it took some experiments to finally comprehend how to display the four digits as if they appear simultaneously. We saved plenty of time in designing the debouncer because we based our debouncer's logic on lab 1's debouncer's logic. Designing the clock module was easier compared to other modules because we had experience with it from lab 1. Furthermore, designing the module to handle button presses and switches was also easier compared to other modules because there were enough examples from lab 1. Lastly, thinking of the circuit behavior in terms of states and designing the circuit in terms of multiple modules really eased the implementation of the circuit.