CS M152A: Introductory Digital Design Laboratory

Lab #3

Stopwatch

Name: Andrew Juarez, Atibhav Mittal

UID: 504-572-352, 804-598-987

Lab Section: 4

Due Date: November 16, 2017

1.1: Introduction and Requirement

The goal of this lab was to design a stopwatch circuit and implement this circuit on the Nexys 3 development board. The inputs to the circuit are two slide switches that control a select line and adjust line, two buttons that control whether to pause or reset the stopwatch, and a 100MHz clock. The output of the circuit is four digits from the onboard seven-segment display, which represents the count of the stopwatch. The left two digits represent the minutes and the right two digits represent the seconds. In the case that pause and adjust are low, the stopwatch was required to increment each second. The seconds digits will count to 59. After 59, it will revert to 0 and the minutes digit will be incremented. The lab manual did not specify what should happen to the minutes digits upon hitting 60, so we decided to revert it to 0 after 59 as well. In the case that the pause signal is high, the display should stop incrementing its count and display its current state until the pause signal is low again. If the reset button is high, then the counters will reset to 0. Debouncing was another feature required to ensure reliable operation of the buttons.

The adjust signal and select signals allow the stopwatch to enter an adjustment mode where the user can select between adjusting the minutes digits or the seconds digits. In particular, if the adjust signal is high and the select signal is low, then the minutes digits are incremented at a frequency of 2Hz and the seconds digits retains its state but are not displayed. If the adjust signal is high and the select signal is high, then the seconds digits are incremented at a frequency of 2Hz and the minutes digits retains its state but are not displayed. One extra requirement in the case that adjust is high is that the seven-segment display will display the digits blinking, instead of a continuous display.

1.2: Design Description

To build this stopwatch, we created 5 different modules, each of which are described below. The first module is the clock_converter, which takes in the hardware clock, and generates 4 other kinds of clocks. The second module is the counter module, which stores the value that needs to be displayed on the seven-segment display. The third one is a modify display module, which controls the blinking of the clock when we're in adjust mode, and the fourth module is the display module that displays the digits on the seven-segment display. The final module is the overall module that instantiates each of these smaller modules and passes in the correct signals from one module to another.

clock_converter.v

Input	Output
• 100 MHz clock (clk) • Reset signal (rst)	• 1 Hz clock (one_hertz_clk) • 2 Hz clock (two_hertz_clk) • 400 Hz clock (fast_clk) • 4 Hz clock (blink_clk)

The aim of this module is to create clocks that run at different frequencies, using a clock that runs at an extremely high frequency. We did this by creating counters for each of the different kinds of clocks we want and then incrementing each of these counters at the positive edge of the fast_clk. Each of the counters, counts up to a different value, corresponding to their respective frequencies, and outputs a clock signal of the required frequency. Each of the counters work so that the clock related to that counter flips at a lower frequency than the rate at which we are incrementing the counters

counter.v

Input	Output
• 1 Hz clock (one_hertz_clk) • 2 Hz clock (two_hertz_clk) • 400 Hz clock (fast_clk) • Pause signal (pause) • Reset signal (reset) • Adjust signal (adj) • Select signal (sel)	• Minutes Ten's digit (minutes_tens) • Minutes One's digit (minutes_ones) • Seconds Ten's digit (seconds_tens) • Seconds One's digit (seconds_ones)

The aim of this module is to maintain the value of the counter for the stopwatch. We maintain 4 different registers, each corresponding to a digit, to store the value of that digit. Once we hit 9 on a one's digit or a 5 on the 10's digit, we increment the digit to the left of that. In the case that the reset signal becomes high, we reset all the digits to 0, and start counting again.

The module uses the 1 Hz clock to increment the counter when the stopwatch is in normal mode. In the case that the adjust signal is high, we want to increment the counter at a frequency of 2 Hz. We do this by using combinational logic, based on the adjust signal to select between the two clocks, and then increment the counter accordingly. When the adjust signal is high, we either increment only the seconds counters or the minutes counters based on the select signal. For instance, if the adjust and select signals are high, we want to only increment the seconds values at a 2 Hz rate. At the positive edge of the clock, we check if the seconds digits are 5 and 9, then we reset it to 0s. Otherwise if the one's digit is at a 9, we reset it to 0 and increment the tens digit. If neither of these cases is true, then we just increment the one's digit for the seconds. It works similarly for adjusting the minutes when the adjust signal is high, and the select signal is low.

Another function of this module was to implement a debouncing circuit. To accomplish this, the example implementation from lab 2 was adopted. A 3 bit register was used to store and synchronize the values of the pause signal. At the positive edges of fast_clk, the pause signal would be sampled. Also, a signal called clk_delay was used to generate a signal one clock cycle behind the fast_clk. At the positive edge of this signal, a rising edge for the pause signal was checked for and if true, the pause_staus signal was toggled.

modify_display.v

Input	Output
• Minutes Ten's digit (minutes_tens) • Minutes One's digit (minutes_ones) • Seconds Ten's digit (seconds_tens) • Seconds One's digit (seconds_ones) • Blinking Clock (blink_clk) • Adjust signal (adj) • Select signal (sel)	• Minutes Ten's digit (o_minutes_tens) • Minutes One's digit (o_minutes_ones) • Seconds Ten's digit (o_seconds_tens) • Seconds One's digit (o_seconds_ones)

The aim of this module is to display the blinking of the clock correctly, which is the case when the adjust signal is high. The underlying algorithm for this display blinking works as follows.

If the adjust signal is low, we output the digits as we receive them, without modifying them. In the case that adjust is high, we set the digits to their actual value for one clock cycle (of frequency blink_clk), and set it to a dummy value (10) to signify that they should be off. We do this using a state machine. The FSM changes state at the positive edge of the blink clock, switching to and fro between the states of display_state and off_state. In the case that select is high, we want to switch off the minutes digits and blink the seconds digits. If at the positive edge of the blink_clk, the current state is display state, we output the actual seconds values to o_seconds_* signals, and the dummy value 10 to the o_minutes_* signals, and set the current state to off_state. When we're in the off_state, we set all the outputs to 10, such that it switches off all the digits. This works similarly for the case when select is low, swapping the o_minutes_* and o_seconds_* signals. This creates the illusion of the digits blinking at the blink_clk frequency.

display.v

Input	Output
• Fast Clock (fast_clk) • Minutes Ten's digit (minutes_tens) • Minutes One's digit (minutes_ones) • Seconds Ten's digit (seconds_tens) • Seconds One's digit (seconds_ones)	• 7 segment display value (segments) • Which digit to illuminate (enables)

The aim of the display module is to take in 4 digits and display them on the seven-segment display decoder. The inputs it takes in are the 4 digits (2 for minutes, and 2 for seconds) and a fast clock. The fast clock is used to cycle through the states displaying different digits so that the illusion of the digits being displayed together is created. We do this using a state machine. The state machine has 4 different states, each one corresponding to a digit. At the positive edge of this fast clock, the state machine moves to the state to display the next digit, working from left to right.

To display each individual digit, we created a module called convert_to_segments that takes in the digit as a decimal, and outputs the values of each of the seven segments. Furthermore, to set the digit, we need to output the correct enables signal, which is a one-hot signal. For instance, if we're trying to display the minutes_tens_digits, we set the enables signal to 0111, so that only the first digit is illuminated on the actual seven-segment display. The first digit in this case will be illuminated with the segments value that we output.

The one special case is that we encoded the digit value 10 to correspond to all offs. Any time a digit is to be displayed as a 10, it means that the entire seven-segment display for that digit should be off.

stopwatch.v

Input	Output
• Hardware Clock (clk) • Reset (reset) • Adjust (adj) • Select (select) • Pause (pause)	• Seven segment display output (segments) • Enable signal for dispaly (enables)

The stopwatch module is the overall module that takes in inputs from the hardware, and outputs the correct digits on the seven-segment display. The stopwatch instantiates all the previously described modules and pipes the outputs correctly. It first instantiates the clock divider module. The 1 Hz, 2 Hz and the fast clock are sent in to the counter module, along with the reset, adjust, select and pause signal. This module then generates the 4 decimal values for the seven-segment display. These digits are sent to the modify_display module. The output from the modify_display module remains the same, unless the adjust signal is high, in which case it makes the seconds or minutes blink as described above. The digits outputted by the modify_display module are sent to the display module. The only task performed by the display module is to generate the correct segments and enables output.

1.3: Simulation and Documentation

clock_converter.v

To test clock_converter.v, a testbench was written to check whether the 1Hz clock and 2 Hz clock were given the correct frequency. The testbench simulated a 100 MHz clock and the outputs were observed in the wave window. For the case of the 1 Hz clock and the 2 Hz clock, the periods were measured directly from the wave window. In the case of the blink frequency and display frequency, the simulator was used to verify that there was a signal toggling at a fixed rate; however, the exact frequency was not important. The visual seen by the human eye was important to displaying the digits at a continuous rate and the blinking were more important. The frequencies for these two clocks were chosen based on varying the frequency and implementing the clocks on the hardware.

counter.v

To test counter.v, a testbench was written that simulated the basic functionality of counting, the reset button, the pause button, and the adjust signal. The first case tested was the reset button. Upon being reset, the counters should reset to 0 and did during the simulation. Next, it allowed the counter to count for 100 timestamps. The output was observed on the console and it was expected that the seconds would display 59, and after would go back to 0 with the minutes digit incrementing. As observed in the following image extracted from ISim's console, this was the case.

```
0 0: 5 5
0 0: 5 6
0 0: 5 7
0 0: 5 8
0 0: 5 9
0 1: 0 0
0 1: 0 1
0 1: 0 2
0 1: 0 3
0 1: 0 4
0 1: 0 5
```

The next case tested for was the adjust signal. After resetting the counters, adjust and select were both made to high. The expectations were that the minutes digit should not be incremented, while only the seconds digit should increment. The output of this test is shown in the following image.

```
0 0: 5 5
0 0: 5 6
0 0: 5 7
0 0: 5 8
0 0: 5 9
0 0: 0 0
0 0: 0 1
0 0: 0 2
0 0: 0 3
0 0: 0 4
0 0: 0 5
```

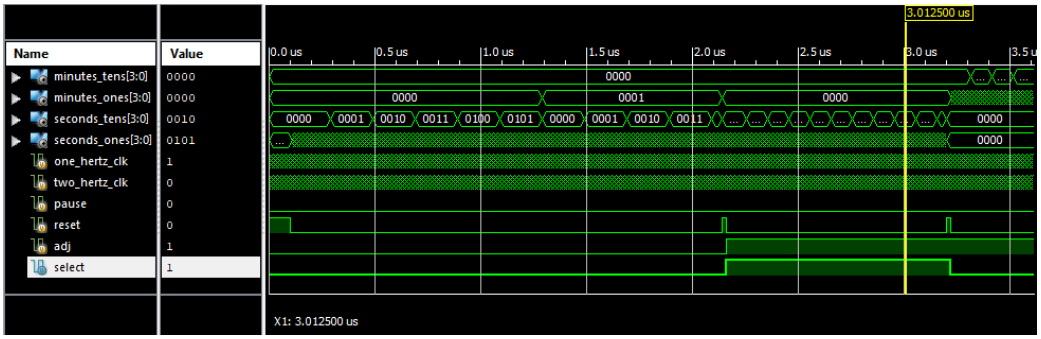
As expected, after 59 the minutes digits were not incremented because in this state, the minutes digits hold their original value. The dual case when adjust is high and select is low was tested for next. In this case, the seconds digit should stop incrementing, while the minutes digit increments at each timestamp. A sample of the output is shown below.

```
5 5: 0 0
5 6: 0 0
5 7: 0 0
5 8: 0 0
5 9: 0 0
0 0: 0 0
0 1: 0 0
0 2: 0 0
0 3: 0 0
0 4: 0 0
0 5: 0 0
```

In this output, only the minutes digits are incremented, and the seconds digits holds its original value. The last test was to check the functionality of the pause signal. Upon pressing the pause button, the counter should stop counting. However, simulating the pause signal in ISim did not imply that the pause feature would work correctly on the hardware. Since there is contact instability for the button, choosing the right frequency to sample the button presses was crucial

and difficult to extrapolate from simulations. As a result, different frequencies of sampling the button presses were varied to find the correct sampling frequency. After running these last tests in hardware, the counter.v module worked successfully.

A section of the waveform for these tests are shown below:

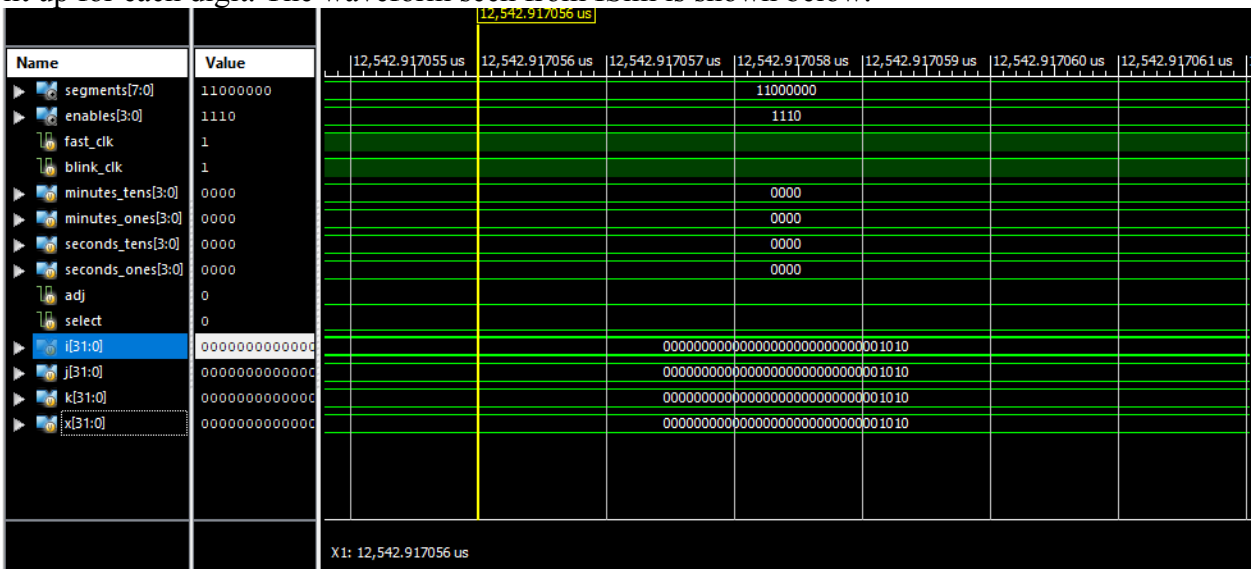


modify_display.v

modify_display was tested by simulating a testbench that checks for the normal case of counting, the case where adjust is high and the select is high, and the case where adjust is high and the select is low. In this module, if adjust is low, then it should simply pass the wires minutes_tens, minutes_ones, seconds_tens, and seconds_ones to the output. When adjust was set low in the simulation, it was observed that each of the inputs was passed to the output. When adjust is set high, it should enter the state machine that causes the display to blink. After tracing through the waveform, when adjust was set high, the output was determined by the state machine causing the display to blink.

Display.v

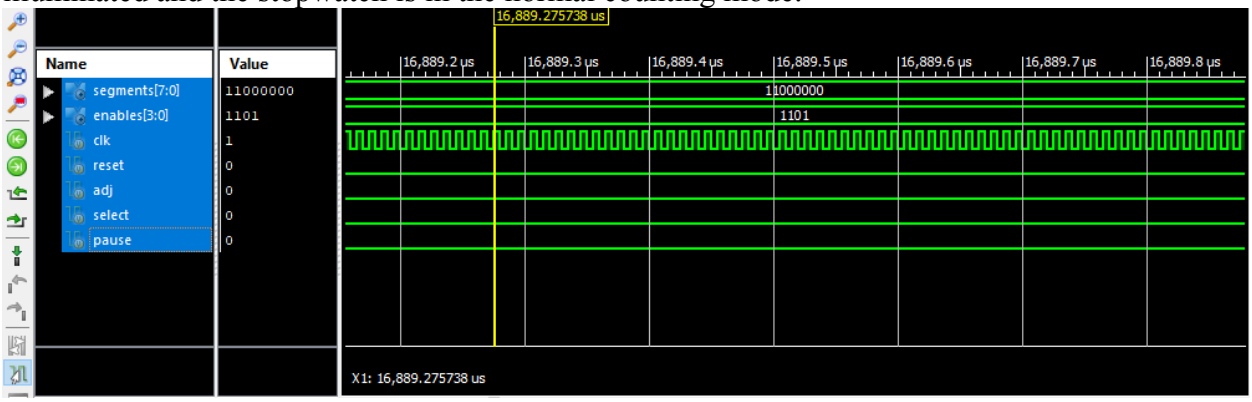
Display was tested by iterating seconds_ones from 0 to 9, iterating seconds_tens from 0 to 9, iterating minutes_ones from 0 to 9, and iterating minutes_tens from 0 to 9. stopwatch.v. In each case the segments were checked by hand with the segments that should be lit up for each digit. The waveform seen from ISim is shown below.



In the waveform, the loops have been traversed so the values of i, j, k , and x are 9. The values of minutes_tens, minutes_ones, minutes_tens, and minutes_ones have been set to 0. The output is the digit 0 for the fourth digit in this case. After observing the output of the simulation and ensuring the outputs followed our logic, the logic was checked by synthesizing the design to hardware. There may have been a logic error in displaying the correct segments so after watching the counter iterate through numbers on the seven segment display, the functionality was verified.

Stopwatch.v

Before testing stopwatch.v, all of the other modules were tested first. Once the other modules' functionalities were verified, a testbench was written for stopwatch that simulated a 100 MHz clock, a pause button, a reset button, adjust signal, and select signal. The testbench handled the case of normal counting, a pause signal going high and back low, a reset going high and back low, and the adjust signal going high. The following image is a capture of the waveform simulated with this testbench. In this section of the waveform, the third digit is being illuminated and the stopwatch is in the normal counting mode.



1.4: Conclusion

The objective of this lab was to program a stopwatch with added functionalities of pause and adjust for the minutes and signals. In the adjust mode of the stopwatch, either the seconds or the minutes had to blink and increment at a faster frequency of 2 Hz. In the normal mode, the clock counters would increase at a frequency of 1 Hz. To meet these requirements, we designed 5 different modules, each of which had a particular task, as described in the Design Description section. The overall module connected these modules to each other correctly and outputted the final result, as a display on the seven-segment display.

We faced a couple of difficulties while creating these modules. One of the main challenges we faced was displaying the blinking of either the second's digits or the minutes digits when the clock was in adjust mode. Initially we had the each of the respective digits blink alternately. However, this did not produce a satisfactory result. We then added another module that would force the digits to be displayed together, and then switch off together. It would alternate between these two states at the blinking frequency we decided. Another challenge we faced was debouncing the pause button signal. We weren't able to figure out how to implement debouncing, but then on further reading the lab manual, we implemented something similar to how debouncing was implemented in Lab 2. Furthermore, we used a 2 Hz clock initially for

debouncing, but this was too slow to get the actual inputs from the pause button. On changing this to a 400 Hz clock, we got the perfect result for the pause button debouncing.

This was an interesting lab, as we got to create an actual stopwatch from scratch, and then see it run on hardware.