CS M152A - Lab 3

Jair Hinojosa, Kyle Romero
904771873, 204747283
Section 5
TA: Fan Hin Hung

Date Performed: 2/20

1.0 - Introduction

Introduced us to the complete FPGA design flow by tasking us with the design and implementation of a stopwatch circuit that would eventually be programmed into the Nexys 3 FPGA Board. The stopwatch uses the board's switches and buttons as a user interface and displays elapsed time on the board's seven segment display.

1.1 - Background

A stopwatch keeps track of elapsed time and often provides the ability to pause, adjust, and reset counting. In order to help us implement this functionality in our digital circuit, the specification explicitly highlights some unique challenges that we will face.

Noise and Metastability

Due to poor metal contact, taking input from buttons on our FPGA Board can often result in a significant amount of noise. A button that is susceptible to this contamination is deemed bouncy.

Furthermore, the term metastability refers to a situation where input signals are asynchronous and the rest of the system is synchronous to some clock. Since two of our inputs come from asynchronous buttons, we must be able to address this issue to implement this lab specification.

Implementation Constraints

Implementation constraints, usually contained in the UCF, are instructions used by the FPGA implementation tools to interface with the hardware on our board. We utilized a UCF to take input or output from our board's hardware. For example, extracting signals from button presses and sending signals to our seven segment display.

Seven Segment Display

A seven segment display provides the simple output for our stopwatch. You are able to create digits on the display by manipulating seven-bit signals to turn lights on and off.

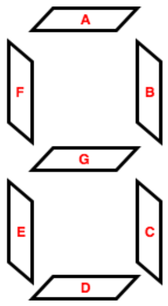

Figure 1.1.1 - The seven segments of a seven segment display.

On our board's displays, we are able to output up to four digits. However, the board only provides us with one line to output display signals. The destination of this display signal is controlled using the anode signal. By quickly manipulating the anode signal, we can change which position digits are written to and create a display that appears constant to human eye's.

1.2 - Design Requirements

The specification outlines that we are to create a simple stopwatch that counts minutes and seconds with its left and right two digits, respectively. For example, after 3 minutes and 45 seconds have elapsed, the stopwatch should read '0345'. Furthermore, the stopwatch must support four functions: reset, pause, adjust, and select. Each function has the following specification:

Reset - The reset button will force all of the counters their initial state, '0000'.
Pause - The pause button will toggle the paused state of the stopwatch. When the stopwatch is paused, the counter won't change. When the stopwatch is not paused it will count up normally.

Adjust and Select

The select signal will select the place that the adjust signal performs its operations on. Both of these signals are implemented using switches.

Select Signal	**Selected Place**
0	Minutes
1	Seconds

Figure 1.2.1 - The select signal and the place that that signal corresponds to.

A high adjust signal indicates that we are in adjust mode. If we are in adjust mode, then normal stopwatch operations are halted and the selected place will increment twice the normal rate. This allows the user to adjust the stopwatch to a new time. Furthermore, the selected place should blink.

2.0 - Design Documentation

We broke our stopwatch up into the following top-level modules: debouncers, segment displays, a clock divider, the time counter, and finally the display driver.

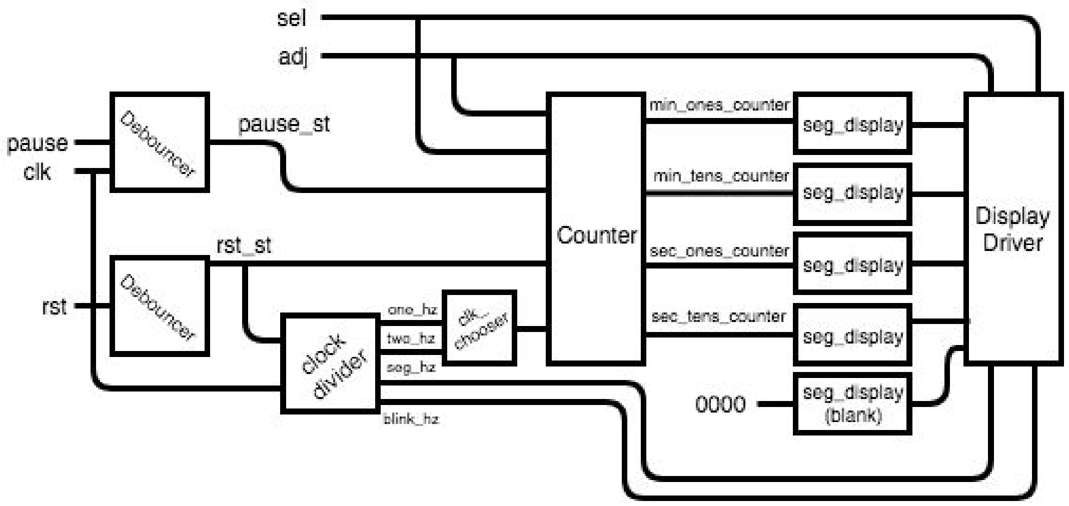

Figure 2.0.1 - The top-level modules of our implementation and their interactions.

2.1 - A Note about User Interface and Undefined Behavior

We developed the following user interface for our lab since the specification was not explicit on such matters.

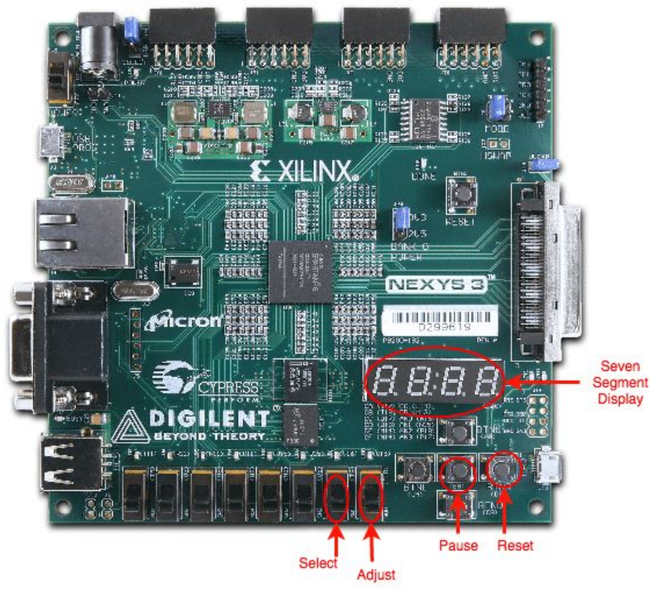

Figure 2.1.1 - The user interface specification for our digital system.

2.2 - Debouncer

Our debouncer is driven by a 20-bit counter that increments when the user is pressing down on the button. If the button is not pressed, the counter is reset to zero. When the counter reaches a specified value (1048575 or 0xfffff, in our case), the state of the button is set high to indicate that a button press was registered. We chose 0xfffff as our threshold by testing many other values for empirical reliability.

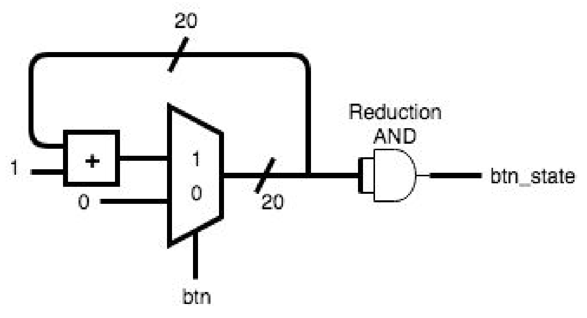

Figure 2.2.1 - A block diagram that demonstrates our debouncer implementation.

2.3 - Clock Divider

Our clock divider uses the system clock to generate four different timing signals: a 2Hz clock, a 1Hz clock, a fast clock (50-700 Hz), and a clock for blinking displays in adjust mode (>1Hz). The 1Hz clock is used to increment the stopwatch and the 2Hz clock is used to increment a selected place in adjust mode. The fast clock is used to quickly cycle the anode signal and create a display that appears constant to the human eye.

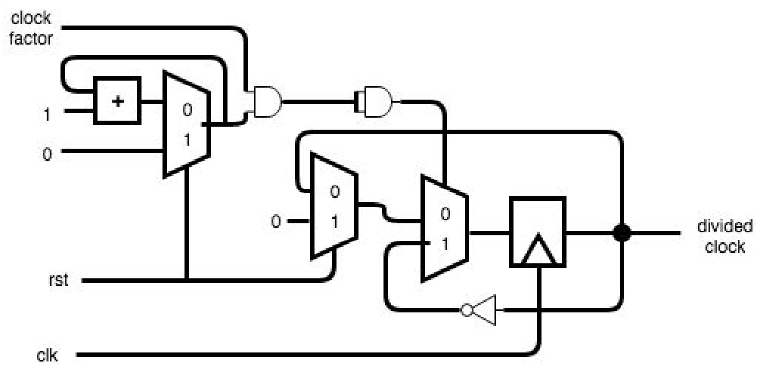

Figure 2.3.1 - A block diagram that demonstrates our clock divider. Note that four of these circuits, with four different clock factors, would be needed to generate our four timing signals.

2.4 - Time Counter

This module stores the tens and ones place counters for both minutes and seconds. As input, the counter takes in: the debounced pause and reset signals, the adjust and select switch signals and the clock that was selected for by the clock selector. These counters feed into the segment displays.

Furthermore, this module ensures the counters are reset to zero when the reset signal is high and that every digit properly carries from the ones to tens place and from the seconds to minutes. Finally, we control the rate of incrementation by selecting the proper clock signal (either 1Hz or 2Hz) using the small clk_chooser module.

2.5 - Segment Display

Our segment display module converts a counter's stored binary value to its seven segment display equivalent. A zero in the eight-bit seven segment display input corresponds to a lit segment.

Decimal Counter Value	8-bit Display Code
0	11000000
1	11111001
2	10100100
3	10110000
4	10011001
5	10010010
6	10000010
7	11111000
8	10000000
9	10010000

Figure 2.5.1 - Conversions between a counter's value to its eight-bit display code in GFEDCBA form (see **Figure 1.1.1**).

2.6 - Display Driver

Our display driver combines the aforementioned modules to implement our stopwatch and drive the seven segment display. Here we ensure that the anode passes the display signal to the proper place on the board and that the correct places on the board are blinking if the user is adjusting the stopwatch.

3.0 - Simulation Documentation

We verified the correctness of our project by looking at several waveforms generated by a simple testbench that we wrote and by empirically testing on the Nexys 3 Board. Because the debouncing module requires signals from the physical FPGA Board, we were unable to test this functionality using only the testbench.

Name	**Description**
General	The stopwatch increments and seconds carry over to minutes
Adjust and Select	Able to adjust both the seconds and minutes, depending on select signal
Pause	Able to pause normal stopwatch functions
Reset	Able to force counters to their initial state

Figure 3.0.1 - The checks that were used to verify the correctness of our digital circuit.

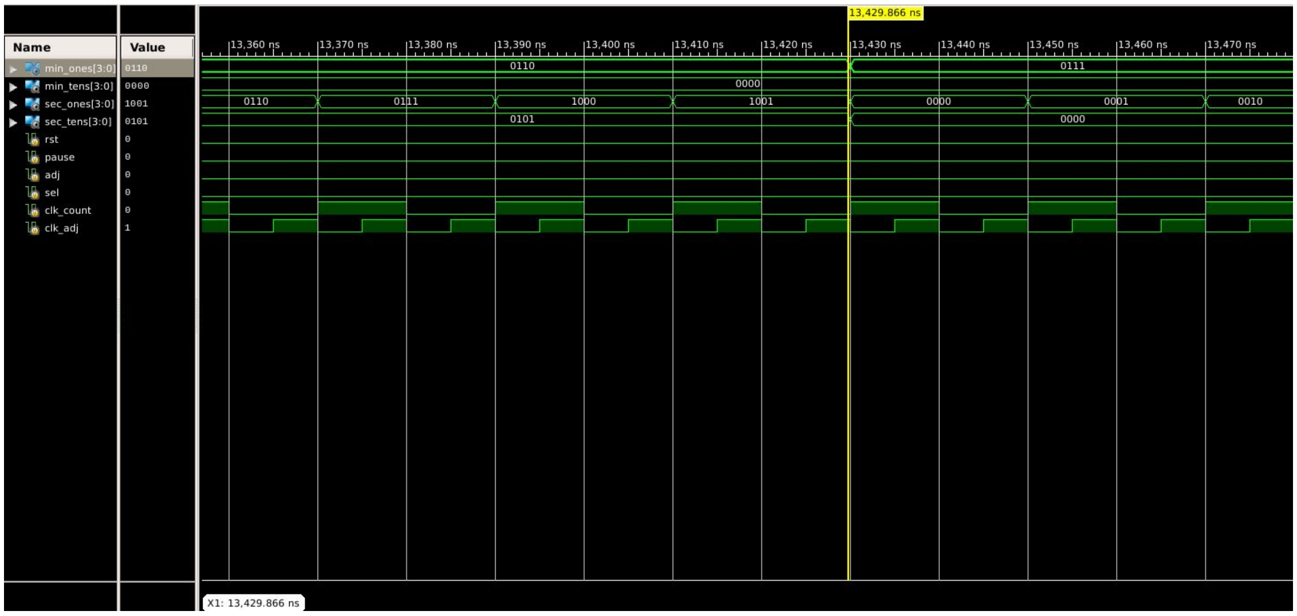

Figure 3.0.2 - The stopwatch counts seconds and values carry over to the minute counter. Note the second value after the yellow flag is 0 and the minute's ones place has been incremented.

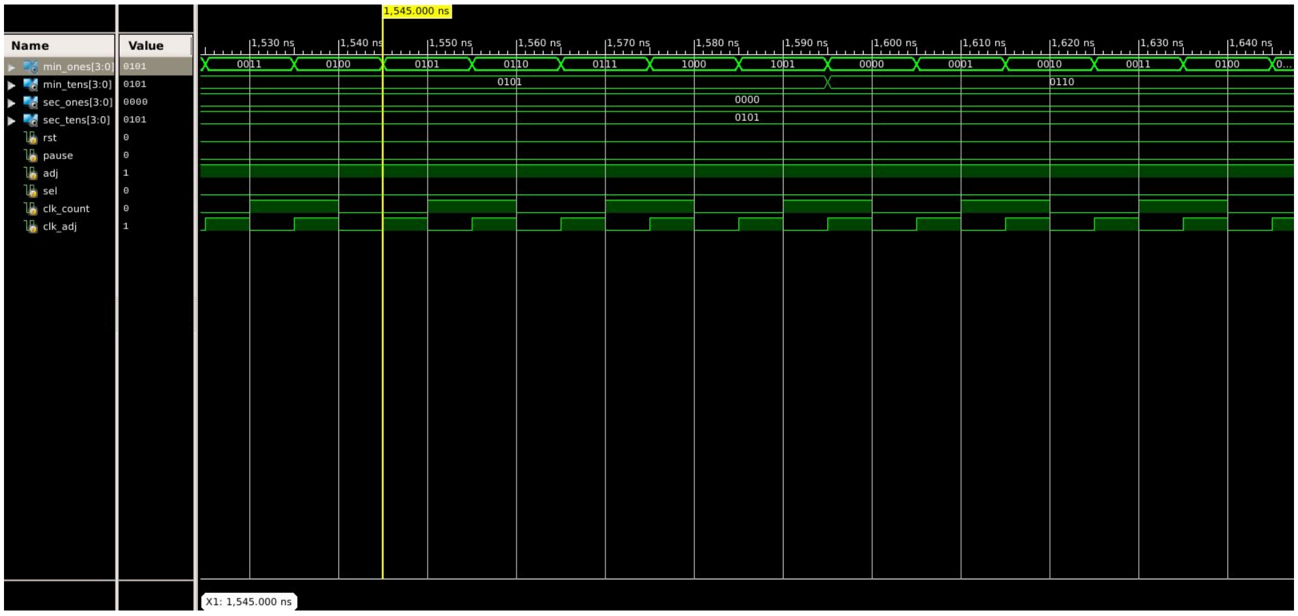

Figure 3.0.3 - The adjust and select switches allow the user to select a new time by increasing the rate of incrementation. Note that the adjust signal is high and the minute's ones place is incrementing quickly.

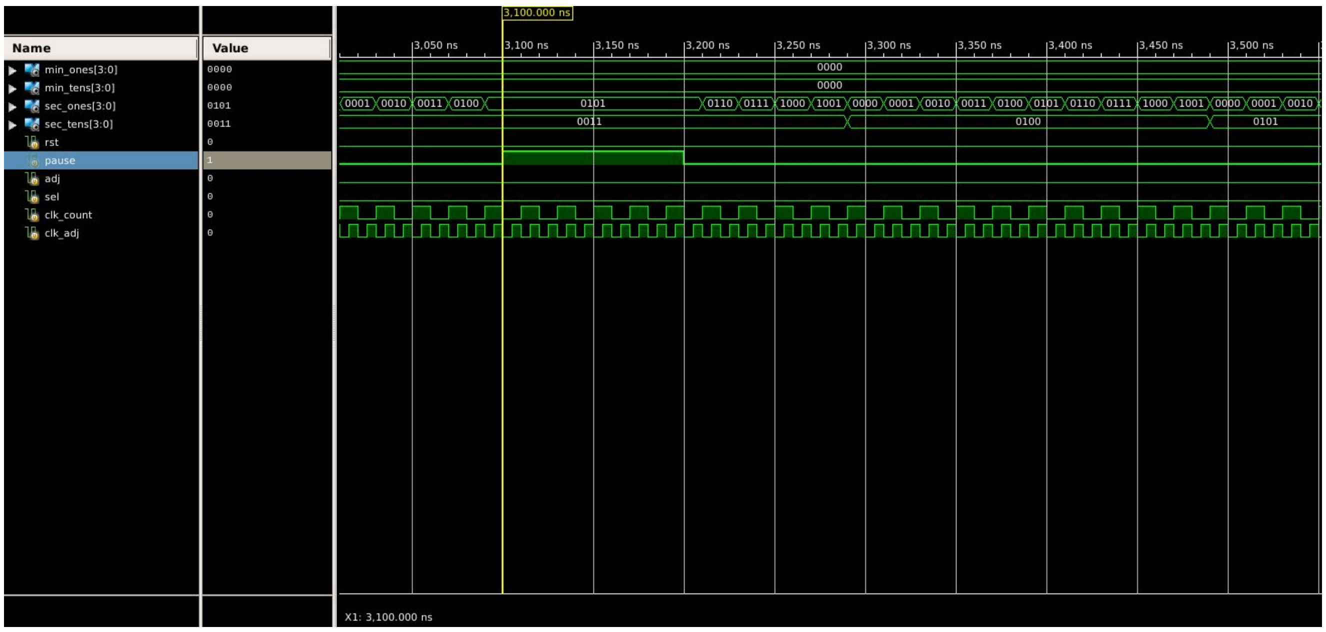

Figure 3.0.4 - The pause button halts normal stopwatch functions. Note that the pause button must be held because the debouncer is not hooked up to the simulation.

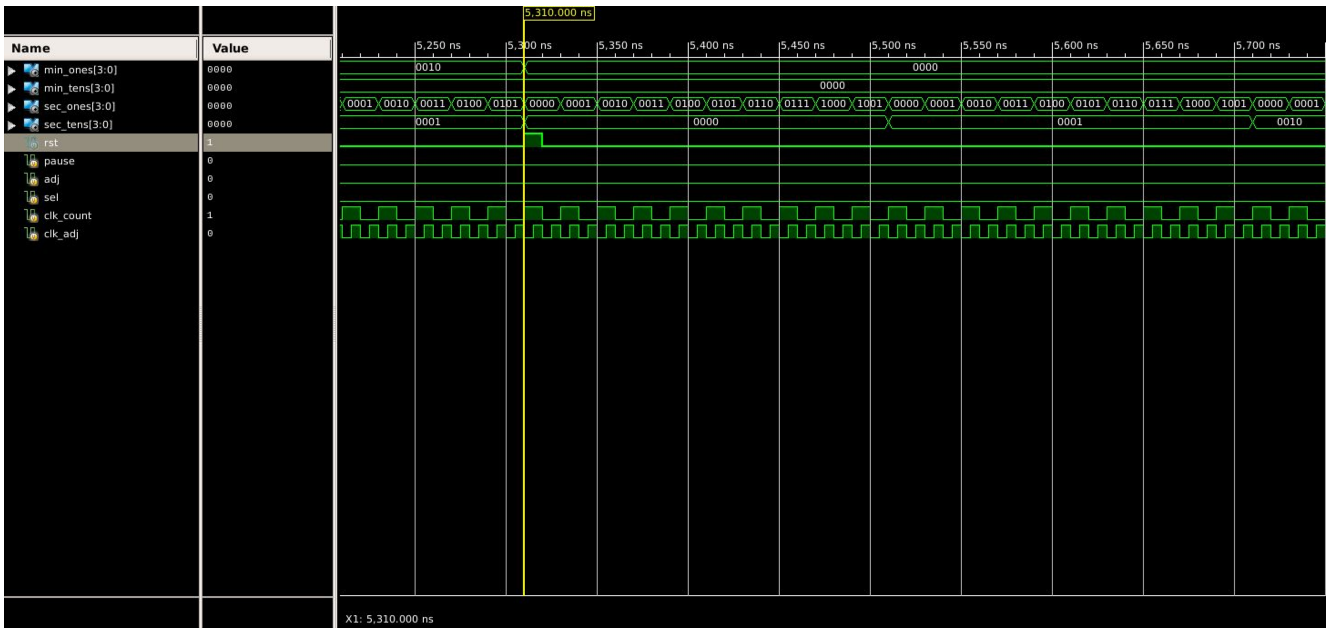

Figure 3.0.5 - The reset button forces all counters to their initial state. Note that all counters are reset to zero once the rst signal is set high.

4.0 - Conclusion

Our design utilizes the buttons and switches on the Nexys 3 Board to drive a stopwatch digital circuit. The button signals must be fed through a debouncer in order to prevent noise and metastability. These signals drive the stopwatch counters which, in turn, drive the seven segment display. A clock divider is used to drive several features of the stopwatch such as blinking the display, incrementing the clock, and adjusting a specified place. Finally, our top-level stopwatch controller connects all the signals to implement our stopwatch.

We encountered several difficulties while implementing this project specification. For example, we had some trouble getting our debouncer to behave reliably. We had to spend some time consulting our TA and trying out different settings in order to get it to behave more reliably. Furthermore, it was a challenge to keep our Verilog project organized while working on a project with so many moving parts. This required us to create separate implementation files for each top-level module and ensure that everything was separated in a logical manner.

We would suggest improving this lab by providing a better specification and outline for how the debouncer should function. Currently, the specification highlights why debouncing is necessary but provides very little for how to implement a debouncer. Therefore, It is difficult to correctly implement a debouncer that works reliably without spending a lot of time trying to figure it out.