CSM152A – Lab 4

Names: Jahan Kuruvilla Cherian; Kevin Xu

Group: Lab 6 – Group 9

UID: 104436427; 704303603

Date Performed: 5/17

TA: Babak Moatamed

1.0 INTRODUCTION

The aim of this lab was to build an entire hardware design from the ground up given the functionality through the specifications in the lab manual. This was tested through the requirement of building a stopwatch that had the ability to count up/down based on the selection mode (0 being the minutes and 1 being the seconds) with the ability to pause and reset the stopwatch. This lab required thinking in terms of Finite State Machines with careful note on the various timing delays that could interfere with a smooth design through the setup and hold times.

In order to tackle the overall module, we broke up the design into subcomponents. This report aims at walking through the steps taken to build out these modules and connect them together in order to produce the final stopwatch.

2.0 DESIGN IMPLEMENTATION

The design for the stopwatch can be thought of as being comprised of a debouncer to account for varying signals from a pressed button, clock dividers to provide the different frequency clocks for each part of the functionality of the display and clock, the counter that performed the logic for storing the counting states for seconds and minutes, segment display to control which bits of the seven segment display lit up to represent the corresponding counter number, and the top module which was used to connect together the components and light up the segment displays by controlling the anode.

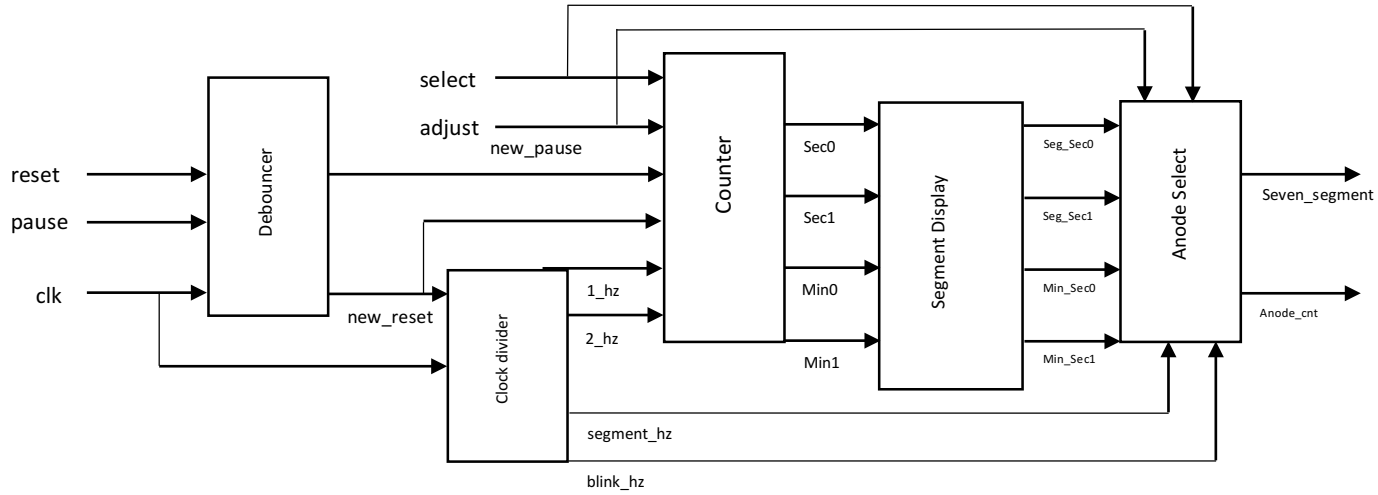

Figure 1: High level block diagram showing the overall design of our implementation of the stopwatch module.

2.1 Debouncer

When the user presses a button, due to the presence of either a spring based or connection based system, the button presents some noise before a stable output to the FPGA. In order for the buttons to consistently work, we had to build a debouncer that essentially returned a valid state of the button once it noticed that the output was constant.

We did this by taking in the button's current input and then doing a high rate sampling using a 16-bit counter. This essentially consistently checked for the value 1 for a long time until it hit the max counter

limit. This guaranteed the output of the new button's state to be a valid one. We performed the deboucing on all buttons in this project, which focused on the pause and reset button.

2.2 Clock Dividers

The next step in the stopwatch design process was to build four distinct clock dividers as follows:

- 1 Hz – This is the frequency of the counter we used to increment the clock's seconds.
- 2 Hz – This is the frequency of the count up and count down modes for either the minutes or seconds based on the select and adjust switches.
- 3 Hz – This is the frequency of the blinking we used for when the adjust was counting either up or down, for the respective selection.
- 1 kHz – This is the frequency of the segment display we used. The anode was cycling through the segments at this extremely high speed in order to make the transitions seem unnoticeable to the human eye.

We essentially used multiple output parameters and created the four clock dividers within one singular module, using 32 bit counters to count to the respective limits for each clock. This is an emulation of the code used for the clock dividing in part 3 of lab 1.

2.3 Counter

The counter is what was used to implement the main functionality of the stopwatch. We broke this design down to using 4, mod-10 counters for each of the seconds and minutes fields. We called them $second_0$, $second_1$, $minute_0$ and $minute_0$ respectively. Within this module we multiplexed the various edge cases through the use of if statements within always blocks. The first case to take care of was when the adjust was not set high. In such a situation, the seconds counter would simply count up and if $second_0$ reached 9, we overflow into $second_1$ and reset $second_0$ back to zero. If $second_1$ got high, then we overflowed into the minute's section and continued on through there. Note that in our design we chose to allow the minutes to reach a high of 99, to account for more than one hour's passing. If the count increased passed 99, or decreased below 0, the counter would wrap the results around.

The second case to consider was if the adjust mode was set to 1, in which case the respectively selected fields (that is if select was 0, then the minutes, else the seconds) would increment at 2 Hz and perform the same overflow checking as before, this time taking care to only affect the selected digits. That is seconds overflowed passed 59 while seconds were being adjusted, then it would wrap back to 0, and not affect the minutes.

Similarly, for the third case if adjust was set to 2, then we would count down the selected fields at 2 Hz. The final nuance of this module was saving the state of the pause button. Essentially we realized that as far as the reset was concerned, a simple click of the button would reset the stopwatch, whereas, with the pause, a click on would suggest pausing and not incrementing the counter, whereas a click off would signal a resume of the counter. The debouncer simply output a stable state of the pause button, which is why we saved the state in the counter and inverted it based on the pressing of the button.

2.4 Segment Sequencer

The segment sequencer was a logically simple module in which, based on the count of the respective counter we would use a switch case to emulate a lookup table such as a ROM, to set the bits as they were meant to. It must be noted that for this design and Nexsys 3 board through Xilinx required that a high bit meant an 'off' of that segment while a low bit mean 'on'.

2.5 Top Module

Using these key modules in our lab, we connected the parts in sequential order within our final top module that represented our finite state machine. We started by debouncing the buttons for safe use throughout the rest of the modules. We then created the four different clocks so that we could use them throughout the rest of the modules. Note that the only modules in which the master clock was used, were the debouncing module for both the reset and pause buttons and the clock dividers module itself. The rest of the modules used the respective clocks they were meant to based on the spec.

We then proceeded to run the counter module to count the respective fields up/down at speeds based on the adjust and select. Using the four counter outputs from this module we, connected each to segment display modules. The final stage of the FSM was controlling the anode of the segment display based on the segment clock, and the blink clock if adjust was set. We simply multiplexed through a 4-bit anode counter that kept switching through the different segments at a high speed. However, if the adjust was set on, we would buffer the display with the blink clock so that the selected fields would turn on and off at a blinking rate that was 3 Hz. The top FSM only had two outputs which were the seven segment displays based on which anode count we were on, and the anode count itself to control the display.

3.0 PROCEDURE

We completed this lab by writing all the Verilog code for all the modules mentioned above, along with a simple test bench. The only module the test bench could not check was the debouncing module, since this directly depended on the Nexsys 3 board itself. Figure 2 and Figure 3 respectively show the results we gathered based on our implementation before placing it onto the board.

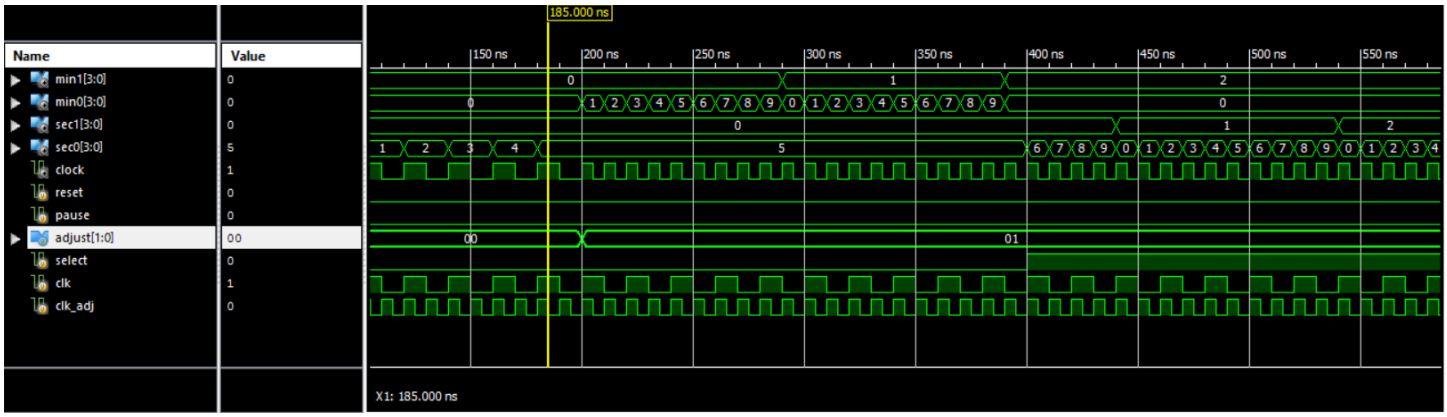

Figure 2: Simulation waveform demonstrating the counting module. Note that the clock output here is the multiplexed clock that selects either the 1 Hz or 2 Hz clock based on the adjust bits. The adjusted clock is noticeably twice as fast as the default clock. Notice that each digit is mod 10 and sec0 (min0) overflows to sec1 (min1), which are the second (minute) digits.

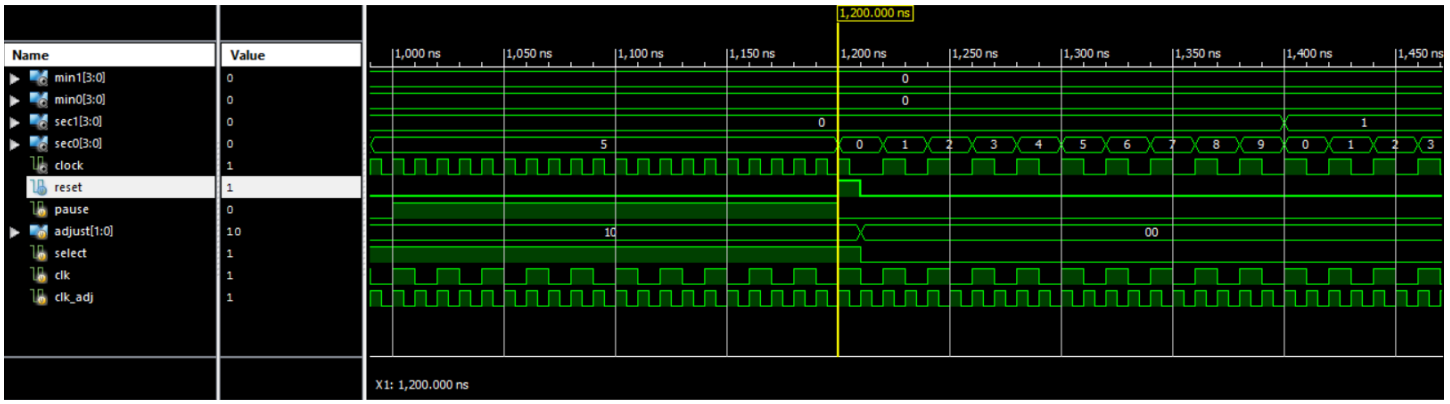

Figure 3: Simulation waveform demonstrating the pause and reset functionality of the design. Notice that the stopwatch digits do not update while pause is 1. Also notice that pushing the reset button at 1200000ns resets the counter value.

Note that the testing for the seven segment display was done through trial and error by displaying onto the FPGA board, until we noticed a recurring pattern wherein 1 meant switching off that segment and 0 meant switching it on.

4.0 CONCLUSION

This lab taught us a lot when it comes from building modules from the ground up, in the process of first designing, coding and testing before implementing. It gave us a better grasp on the various use cases within Verilog in Xilinx, and taught us about the importance of correct timing and debouncing in real time hardware applications.