CS M152A: Introductory Digital Design Laboratory
Lab 3

Elton Leong
Anthony Lai

Introduction and Requirement

The goal of Lab 3 was to design a stopwatch on the FPGA board. This stopwatch had several buttons and switches inputs that controlled the state of the stopwatch. The digits of the stopwatch were then displayed through the on-board seven-segment display. By going through the complete FPGA design workflow, we designed the stopwatch circuit and implemented it on Nexys™3 Spartan-6 FPGA Board.

Functionality

Specifically, the stopwatch we designed had 2 switches and 2 buttons as inputs. The two switches, **SEL** and **ADJ**, were used to adjust the timer of the stopwatch. The **SEL** switch controlled the "selected digit", the digit to be adjusted in the *adjusting mode*. When **SEL** was set to logic low (0), "minutes" were being adjusted. Whereas when **SEL** was set to logic high (1), "seconds" were being adjusted. On the other hand, the **ADJ** switch was used to toggle the *adjusting mode*. When **ADJ** was set to 0, the stopwatch behaves normally; when **ADJ** was set to 1, the normal functionality of the stopwatch stopped, and the seven-segment display started blinking while the "selected digit" increased at a rate of 2Hz.

The two buttons, **RESET** and **PAUSE**, were used to control the state of the stopwatch. When **RESET** was pressed, the stopwatch counter would be reset to 00:00 and restarted the basic functionality of the stopwatch. When **PAUSE** was pressed, the stopwatch's counter was paused, and it only resumed when the **PAUSE** was pressed again.

Design Description

The top level design of the StopWatch module is shown below in Figure 1.

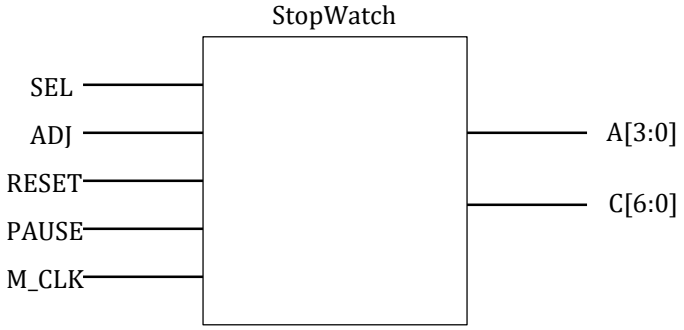

Figure 1: The inputs and outputs of the top level StopWatch module

In addition to taking the 2 buttons and 2 switches as inputs, the StopWatch module also takes in the master clock signal, M_CLK, from the internal clock of the FPGA. The StopWatch module has two outputs, A[3:0] and C[6:0], to control the on-board seven-segment display (A[3:0] controls the digit to be displayed while C[6:0] controls the number to be displayed).

To implement the StopWatch module, we decided to split the design into five separate modules: ClockDivider, SwitchDebouncer, ButtonDebouncer, TimeToDisplay, and DisplayTime. This implementation allows us to test each module separately and enhances modularity.

Interaction Among the Modules

ClockDivider's outputs are used for all of the other modules as the clocks that control their operation. The debouncers' outputs are fed into TimeToDisplay, which is the main controller for the stopwatch. TimeToDisplay's outputs, which are the individual digit counters and whether the stopwatch is in *adjusting* mode, are fed to DisplayTime. DisplayTime's outputs are directly connected to the output of the top level StopWatch module.

Design Specification for Each Module

ClockDivider Module

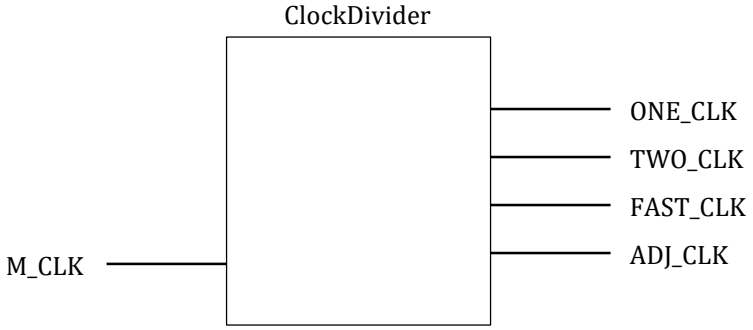

Figure 2: The inputs and outputs of the ClockDivider module

As shown in Figure 2 above, the ClockDivider takes in one input, M_CLK, and has four outputs, ONE_CLK, TWO_CLK, FAST_CLK, and ADJ_CLK. The master clock with frequency of 100MHz is connected to the M_CLK as input of the ClockDivider module. Using this master clock signal, the ClockDivider module outputs four different clocks, ONE_CLK, TWO_CLK, FAST_CLK, and ADJ_CLK with frequencies of 1Hz, 2Hz, 625Hz, and 5Hz, respectively. These four different clocks are then used in other modules to control the specific behaviors of the StopWatch. The actual divider is internally implemented as four different modulo counters. When the integer counters wrap around, the clocks are set to high for the duration of the next cycle of M_CLK.

SwitchDebouncer and ButtonDebouncer Modules

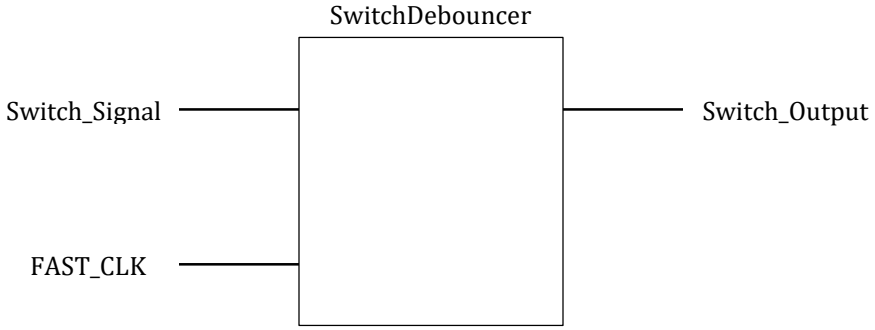

Figure 3: The inputs and outputs of the SwitchDebouncer module

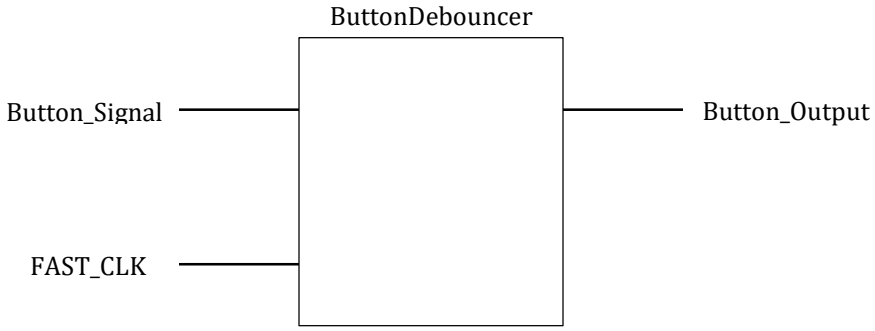

Figure 4: The inputs and outputs of the ButtonDebouncer module

As shown in Figure 3 and Figure 4 above, the SwitchDebouncer and ButtonDebouncer takes inputs from either the switches or buttons and the FAST_CLK from the ClockDivider module. The main purpose of the two debouncer modules is to filter out the noises from switches and buttons due to poor metal contacts. To accomplish this, our codes sample the signals from the button and switches at a frequency lower than that of the noises. The concept is illustrated below in Figure 5.

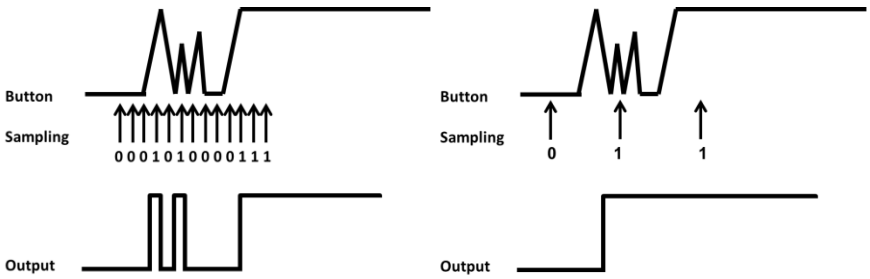

Figure 5: Sampling the signals from the button at different rate produces different output

After few trials and errors, we determined that the best sampling rate is the frequency of the FAST_CLK. Referencing the implementation from Lab 1, we were able to implement the debouncer modules successfully and picked up useful signals from the buttons and switches.

TimeToDisplay Module

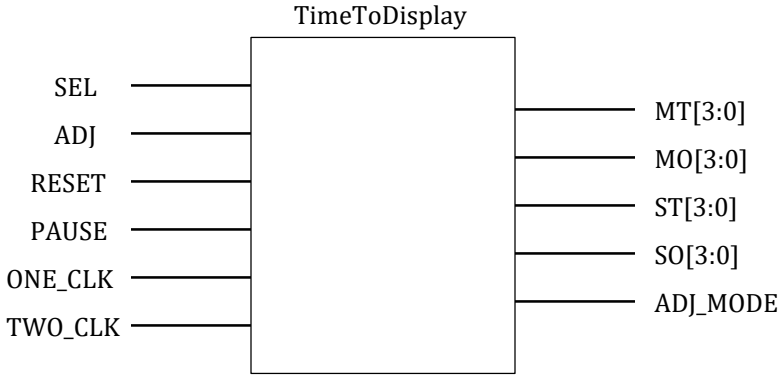

Figure 6: The inputs and outputs of the TimeToDisplay module. The buttons and switches inputs, SEL, ADJ, RESET, and PAUSE are outputs from the debouncer modules.

The TimeToDisplay module is the heart of the StopWatch module. It takes in inputs SEL, ADJ, RESET, and PAUSE from the deboucner modules discussed above as well as the ONE_CLK and TWO_CLK signals from the ClockDivider module. It then outputs MT[3:0], MO[3:0], ST[3:0], and SO[3:0] which represents minute's ten's digit, minute's one's digit, second's ten's digit, and second's one's digit, respectively. It also has an additional ADJ_MODE output pin to signify whether the stopwatch is in *adjusting* mode (Figure 6).

The implementation of the TimeToDisplay module is as follows: it first checks whether the stopwatch is in *adjusting* mode. If the stopwatch is in normal mode, it increments SO[3:0] on the positive edges of ONE_CLK; otherwise, it increments SO[3:0] on the positive edges of TWO_CLK. When SO[3:0] becomes 10, the module resets it to 0 and increments ST[3:0] by 1; when ST[3:0] becomes 6, the module resets it to 0 and increments MO[3:0] by 1 (and similarly for MO[3:0] and MT[3:0]). For the RESET and PAUSE functionalities, the module achieves the design requirements by simply resetting all counter to 0's on the next positive edge of ONE_CLK, upon receiving the RESET signal, and it halts the increments on all digits, upon receiving the PAUSE signal. If it receives the PAUSE signal again, then the increments on all digits begin again.

DisplayTime Module

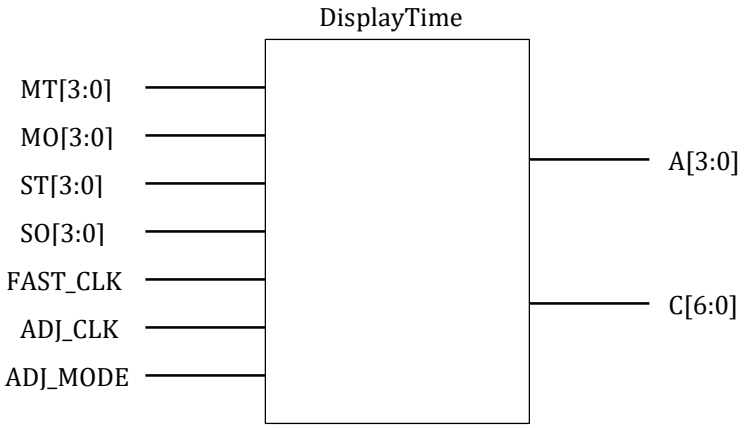

Figure 7: The inputs and outputs of the DisplayTime module

The DisplayTime module takes the binary inputs from the TimeToDisplay module and converts them to seven-segment encodings (Figure 7). It will then output the digits, one at a time, to the seven-segment display. The next digit to display is output at every positive edge of FAST_CLK, so that humans looking at the display cannot detect its refresh rate. The output C[6:0] specifies the seven-segment displaying patterns whereas the output A[3:0] specifies the digit to be displayed. The binary-to-seven-segment conversion is shown below (Figure 8).

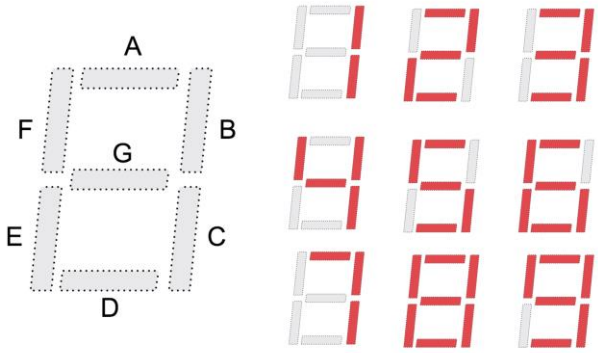

Figure 8: Binary-to-seven-segment conversion

In addition to converting the digits, the DisplayTime module also checks whether the stopwatch is in *adjusting* mode or not via the ADJ_MODE input pin. If the stopwatch is currently in the *adjusting* mode, the DisplayTime module will control the display so it blinks at the frequency of the ADJ_CLK.

Simulation Documentation

In ClockTest.v, there is a simple test bench that pulses the clock at the same rate as the Nexys 3 board's master clock. Running this bench for a simulated one second allows us to test whether the ClockDivider module functions properly by checking whether the four outputs of the module are set to high at the proper times they are expected to.

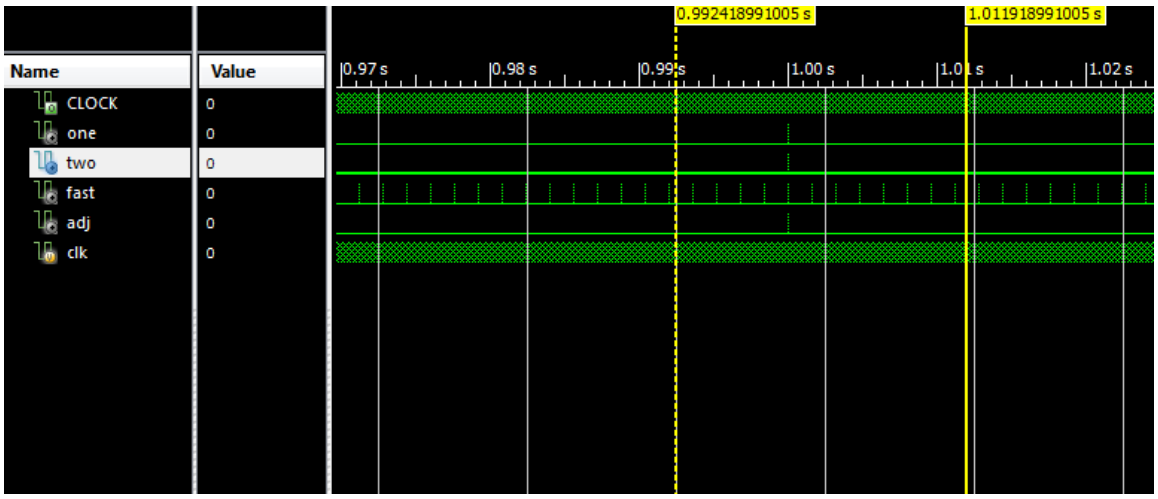

Figure 9: Clock divider simulation waveform

As seen in the above waveform (Figure 9), ONE_CLK, TWO_CLK, and ADJ_CLK are all set to high at the 1.00s mark, which is expected. FAST_CLK is set to high at the expected intervals, indicating that the module works properly when simulated.

In StopwatchTester.v, the functionality of the **RESET** button is tested. After 1.1 seconds have passed, **RESET** is pressed and then released. The expected behavior is that at the 2 second mark, the clock resets to 00:00 instead of continuing to 00:02.

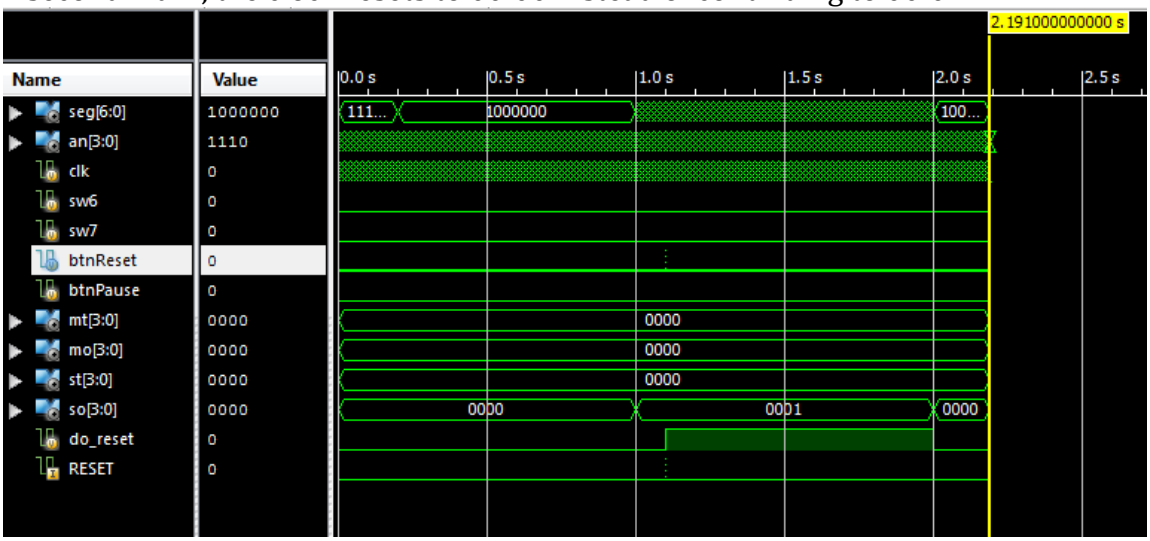

Figure 10: Reset button simulation waveform

As seen in the above waveform (Figure 10), the **SO[3:0]** counter resets from 1 second to 0 second at the 2 second mark, as expected.

Conclusion

To enhance modularity and be able to test individual modules in ISim, the StopWatch module was split into 5 different modules. ClockDivider divides the master clock from the board into four different clocks that operate at 1 Hz, 2 Hz, 625 Hz, and 5 Hz. ButtonDebouncer and SwitchDebouncer sample the buttons and switches at a lower frequency than that of the master clock so as to be able to disregard noise masquerading as a button press or a switch flip. TimeToDisplay controls the digit counters, handles *adjusting* mode, and the **PAUSE** and **RESET** functionalities. DisplayTime controls the display based on the digits fed to it by TimeToDisplay and *adjusting* mode being active or not.

Bugs were not found during simulation. The bugs that were found were discovered when testing behavior on the Nexys 3. These bugs were caused by delivering the A[3:0] and C[6:0] outputs either with the order backwards, or the bits inverted, which caused incorrect updating of the seven-segment display. There were difficulties with simulating the StopWatch module, though, because it took several minutes to simulate one second of the module running. Fortunately, the TimeToDisplay module behaved as expected, which minimized the amount of times the StopWatch module had to be simulated.