Lab 3 - Stopwatch Implementation

Fnu Shiva Sandesh, Junyoung Kim, and Brandon Tai

May 22, 2017

M152 A - Miodrag Potkonjak

TA: Brinda Alluri

1. Introduction and requirement

In this lab assignment, we were required to design a stopwatch on the Nexys™3 Spartan-6 FPGA board, hardware design, by using our knowledge of finite state machine that we learned from our CS M51 class and previous lab assignments. The stopwatch needs to be able to count seconds and minutes, adjust its values based on selection mode, pause the count, and reset the stopwatch to zero. The digits of the stopwatch required to be displayed through the on-board seven segment display of FPGA board, and the pause and reset button also need to be implemented on the board. In order to complete the design, we had to use four internal clocks with different frequencies, all derived from the master clock. There are couple of design challenges that we faced when implementing this lab. The first one was being able to display all the four digits with active second count. The value displayed on the other three digits changes based on the value of the second count, which comes naturally from the clocks we use everyday. Another interesting design challenge is implementing the adjust button which results in Stopwatch being stopped and the 'selected' value blinks and increases at 2Hz. The following tables describes the design requirements for the lab.

Table 1: Description of the Select Mode (1 of 2) and Adjust mode (2 of 2).

SEL	Selected
0	Minutes
1	Seconds

(1 of 2)

ADJ	Action
0	Stopwatch behaves normally
1	Stopwatch stops and *'Selected'* increases at 2Hz

(2 of 2)

Lastly, the final challenge was handling the meta-stability of the buttons. Meta-stability exists because of the asynchronous nature of the button and slider switch input. To ensure reliable operation in digital circuits, the input to a register must be stable for a minimum time before the clock edge and for a minimum time after the clock edge, which is the signal timing requirements in its simplest form. In synchronous systems, the input signals must always meet the register timing requirements. Yet in the case of the buttons and slider switches, the signal may come at any time, causing unpredictable outputs to the other modules connected to the signal. To handle it we simply used a counter to check if the button was pressed for minimum of 10 ms. Thus, enabling us to process the input from the FPGA board correctly as required by the description of the lab.

2. **Design description**

The design decisions for the stopwatch were centered to make the code easier and flexible for future usage. In order to fulfill this, the task was divided into four main components: the clock divider, counter, seven segment display logic and debouncer respectively. Since we needed four different clocks frequencies, we converted the master clock of 100 M*Hz* into a 256 H*z* clock. We did this by creating a counter which counts up to 62500 before toggling the state of a secondary clock and resulting in clock of 256 H*z.* After this we used the 256 H*z* clock and converted it into a 1 Hz, 2Hz, and 4 Hz clock. Following code snippet depicts the conversion of 100 M*Hz* into 256 H*z* clock.

```
always @(posedge(CLK)) begin
 if (COUNT_256Hz == 19'b1111010000100100) begin
      CLK_256Hz = ~CLK_256Hz;
      COUNT_256Hz = 19'b0;
   end else begin
      COUNT_256Hz = COUNT_256Hz + 1;
   end
end
```

The purpose of the 1 Hz clock was to act as the second counter and all the other digits were based on the variable used for counting the seconds count. This means that we increase the tenth position of the second count after every ten seconds and increase the minute count when the tenth digit of the second count reset from 5 to 0, which will happen every one minute. The 2 Hz clock was used for adjusting the selected digits so that they will be incremented twice per second. Lastly, the 4 Hz clock was used as blinker frequency for the selected digit. For the sensitivity list we used the positive edge of the 2 Hz clock and internally decided if to use 1 Hz or 2 Hz based the state of the input.

In order to successfully implement the stopwatch, we needed additional constraint such as reset, adjust, and pause. Since these were the inputs from the FPGA board so all we have do was to add these additional constraints to our loop. So we made flags for each input, and checked each of these flags before the decision of whether to use the 1 Hz clock or 2 Hz clock and more importantly which of the four digits needs to changed. That is to say that if the adjust flag is set to 1 then we look at the select input to decide which of the digits needs to changed. However, at any stage the user might pause the timer or reset it to zero, so we added a another condition to our loop structure to check for those conditions as well. As mentioned earlier this required changing the frequency used. The following state diagram presents a high level depiction for which frequency was being used with different set of inputs.

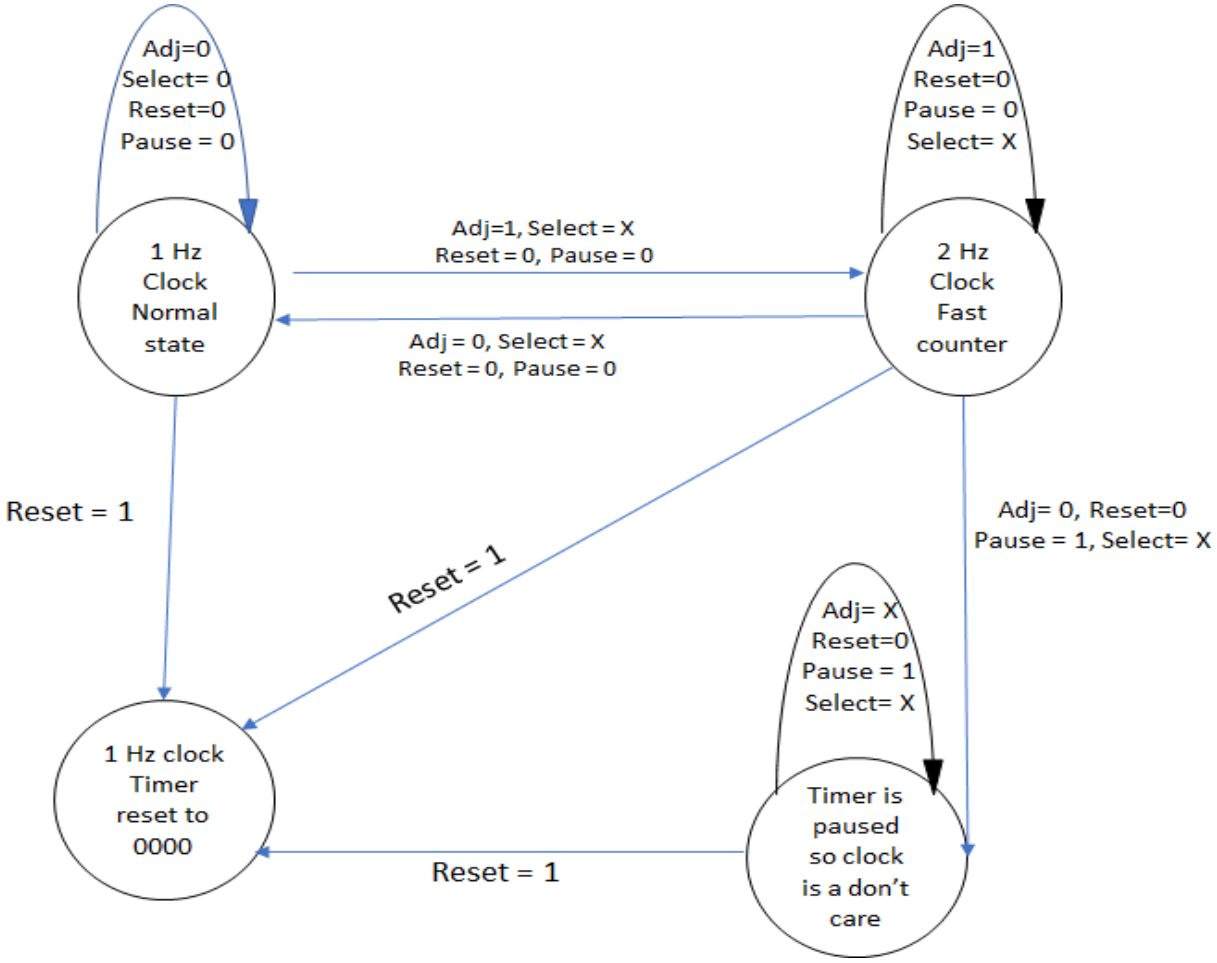

Figure 1: Depiction of logic for switching the clock frequency

Having done that we know we had to implement the seven segment display logic. We used the faster (256 Hz*)* clock to assist in the multiplexing of the seven-segment display. Since the display is designed in such a way that all four seven-segment displays will show the same digit if no additional circuitry it used. For this reason, we cycled through the four digits using a much faster clock so the human eye sees four digits. For efficiency, we made a function to convert a 4-bit binary number into the boolean 7 segment display representation. Since we were using 4 separate variables, each corresponding to a digit on the seven segment display, we had to concatenate a zero in front of the digit corresponding to the tenths position for both minute and second because those digits were 3 bits instead of 4 bits.

The last part of the lab was the implementation of the debouncer to ensure the proper behavior if the reset or pause button is not pressed properly. For this we used a counter to check if the button was pressed for the certain amount of time and thus accordingly we set the value of the reset and pause signals. The following code snippet depicts it. The value of the counter is checked later on in the code before changing the reset and pause signals. The counter goes up to 20'b10000000000100000000 which roughly corresponds to 10 ms. The following code snippet describes it.

// For the RESET button *if (RESET == 0) begin* *RESET_counter = 0;* *end else begin* *RESET_counter = RESET_counter + 1;* *end*	*// For the PAUSE button* *if (PAUSE == 0) begin* *PAUSE_counter = 0;* *end else begin* *PAUSE_counter = PAUSE_counter + 1;* *end*

3. **Simulation documentation**

We did the simulation directly on the FPGA board in order to understand how do seven segment display and circuit board work. At first, we started simulating the code with only master clock that has 256 Hz, and then we observed if the clock was counting the minutes and seconds; however, the seven segment display turned fully on as 8888, and didn't count at all. We quickly realized that we had some logic error to turn the display on. After carefully reading the lab-manual again we found the logic error and fixed it based on the figure below,

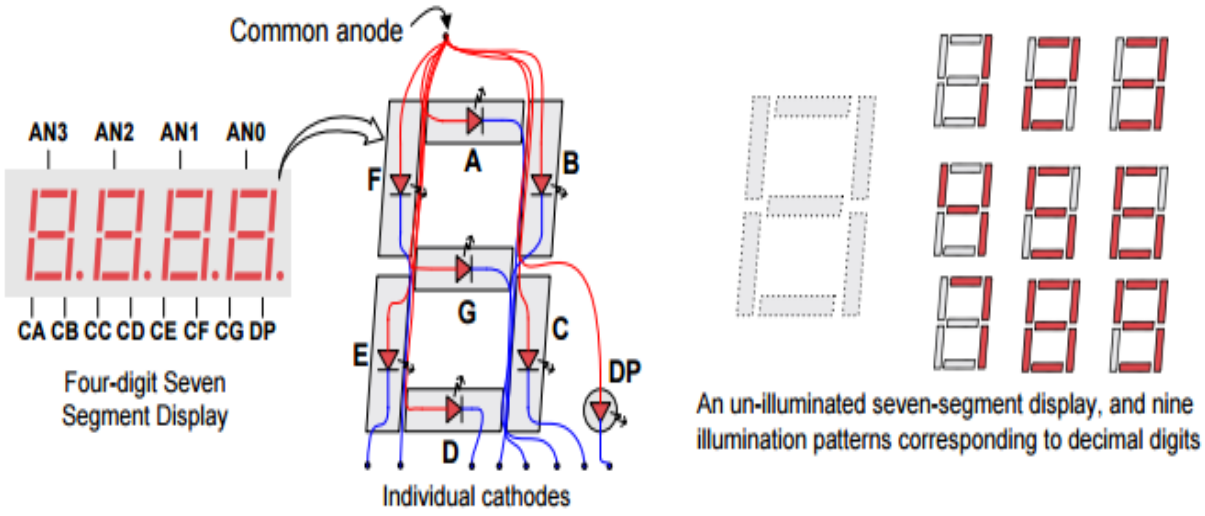

Figure 2: Seven Segment Display Letter

After we fixing the display, we realized that the counter counted second little slower than actual time. It was counting 8 seconds during 10 second time period. We asked Brinda the reason and she told us that since the logic of code is correct we should not worry about the slow counting because it can be caused by the wire connection or some other minor reasons.

Implementing pause and reset buttons were relatively easier than setting up the count because we just needed to implement basic logic of if else statements to control the master clock to stop or restart when the button are activated.

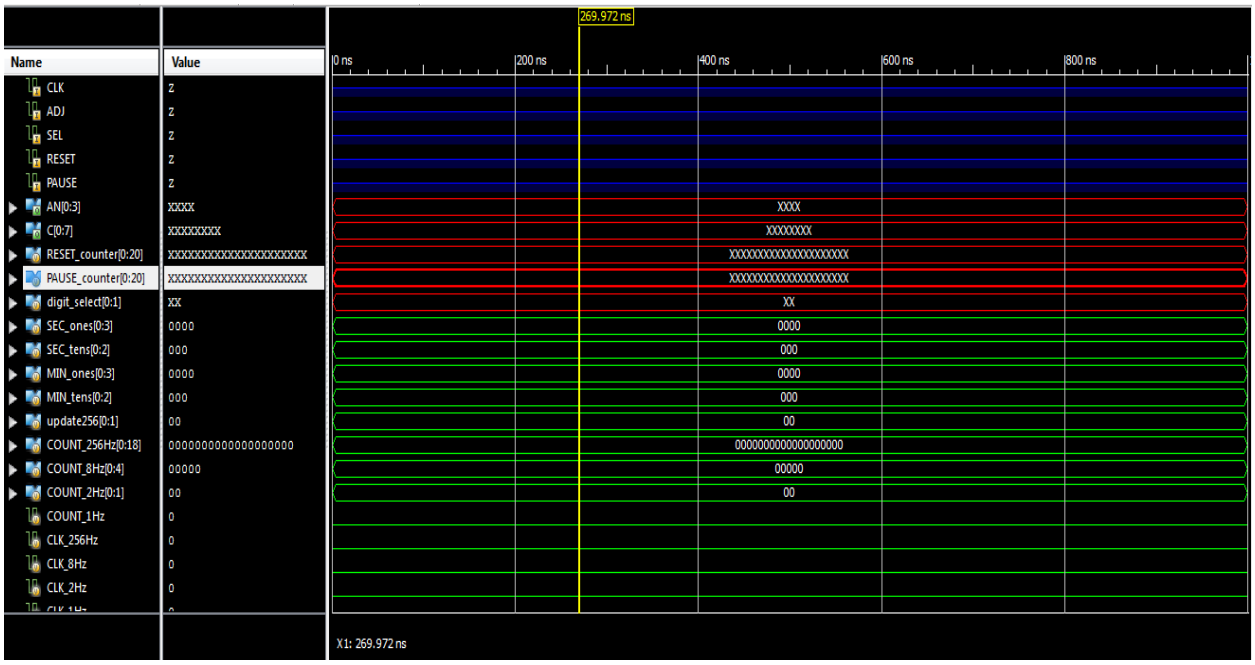

Figure 3: Waveform of stopwatch

From the above figure, we could see that the waveforms are not meaningful because as we mentioned above, we directly implement the logic into FPGA board instead of implementing on the test bench because we wanted debugging with the hardware not the software since we already have done couple of assignments debugging solely on the software.

4. **Conclusion**

During this lab we encountered a number of difficulties and majority of them were related to the syntax of verilog. Initially, we had having two separate always block, one for the normal functioning of the stop watch and the second for various configuration of the inputs, adjust and select respectively. This resulted in multiple syntax errors because we were initializing the same variables in two separate always sections. We were primarily doing it for code coherence and enhance code readability. However, the problem could not be resolved without doing all our calculations within a single always block. Although it lead to code duplication, however, the code was fully functioning. The other major problem that we faced was making the LEDs blinks at a frequency that was recognizable by human eye and at the same time showed the linking effect. Ultimately, after various trials and errors we were able to narrow 4 H*z* frequency to the optimal frequency. The last challenge was to implement the code to handle debouncer. For this we made a simple counter that counts up to 10 ms before deciding whether the button was pressed or not.

Overall, we feel that lab was certainly challenging however, it was a good insight into how real life objects can be simulated using FPGA. Thus, it was a learning experience, and helped us in understanding the UCF file in its interaction with the FPGA. We learned which pins are used and how the naming convention works between the UCF file and the verilog file.