***Lab 6 – Implementing a 4 Digit Stopwatch***

## Review Item Comments Points (max)

1. Prelabs from each student complete (5)

and thoughtful

1. Introduction effectively presents the (5)

objectives and purpose of the lab. Methodology gives enough details to allow for replication of procedure.

1. Discussion opens with an effective (5)

statement on the goals of the lab, backs up the statement concerning appropriate findings, provides a sufficient and logical explanation for the statement, addresses other issues pertinent to the lab.

1. Results open with an effective statement of (5) overall findings, presents visuals clearly

and accurately, presents findings clearly and with sufficient support. You MUST

include screenshots of the test bench results for each part of the lab.

Conclusion convincingly describes what has been learned in the lab.

1. Other: (10)

References are included.

Tables and figures are formatted. Grammar and spelling are correct

Comment Blocks for ALL Verilog modules are filled

in with students names and module description/purpose Report is written clearly and to the point.

Overall, the team...

* + has successfully demonstrated what the lab was designed to teach
  + demonstrates clear and thoughtful scientific inquiry
  + has accurately measured and analyzed data for lab findings

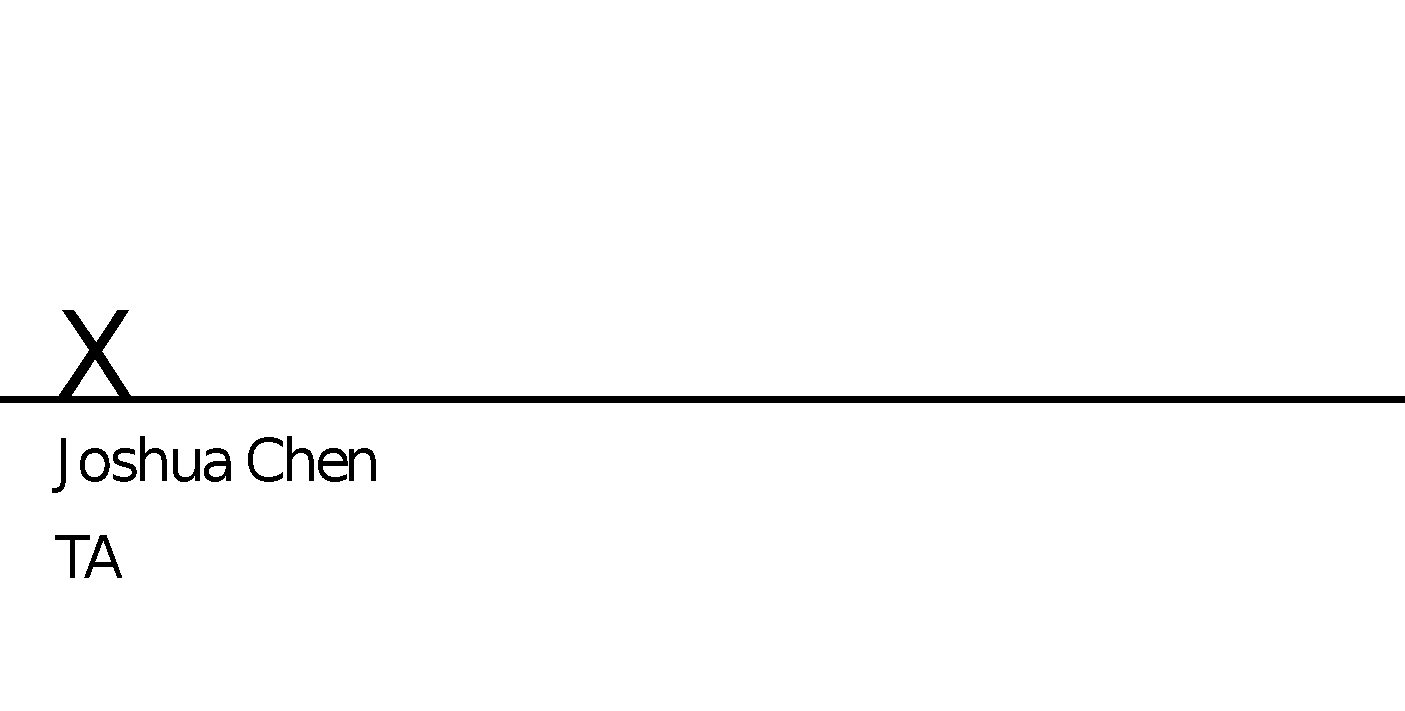
Total: (30)

|  |
| --- |
| EE260 Lab Lab Number  **Using Sequential Logic** |

**Submitted by:**

**doesthatmakesense**

Grade:



|  |  |
| --- | --- |
| Team Members: | Josh Mellinger  jmellin@hawaii.edu |
|  |  |
| Date Experiment Performed: | 3 April 2018 |
| Date of Submission: | 15 April 2018 |
|  |  |
| EE260 Lab Section | Section 001 |

|  |
| --- |
| Introduction |

Introduction effectively presents the objectives and purpose of the lab.

The objective of this lab was implement a digital stop watch using Verilog for an FPGA. The purpose of this lab was to gain additional experience using the onboard clock, specifically reducing the 100 MHz default frequency. An addition purpose was learning to implement sequential logic in Verilog code.

|  |
| --- |
| Methodology |

Methodology gives enough details to allow for replication of procedure. (You may assume lab assignment is available. You do not have to regurgitate the lab instructions).

To start this lab you will need to write a module that reduces the frequency of the 100 MHz clock. There is some skeleton code provided, it will need to be modified initially to reduce the clock to 10kHz. Then using that 10kHz clock, reduce the frequency again to 10 Hz. Remember that the clock is an input into your design source, but the clock needs to be assigned in the constraint file to actually work. To test it, assign the output in the constraint to one of the LEDs on the FPGA, it should blink 10 times a second if your clock is reduced correctly.

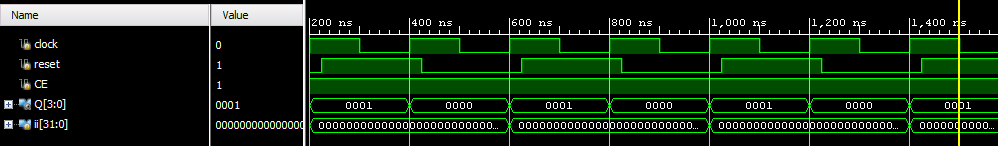
You will need to then write a module that counts up, increasing by 1 on the positive edge of the 10 Hz clock. Using binary, the clock should count up from 0000 to 1001 (0 to 9) then start over. Use a case statement or a series of if/else statements. This first module can serve as the count for your tenths of a second part of the clock. The seconds module can also go up to 10, but remember that there are only 60 seconds in a minute, so the minutes module should only go up to 5 (0 to 5).

Perhaps the most challenging portion of the lab is getting the clock to count up correctly. The reset signal from the less significant digit should be sent when it is at 9. Also, the synchronous reset should also be coded in with the @(posedge) to ensure that all of the digits reset at the same time to avoid unsynced times on the clock.

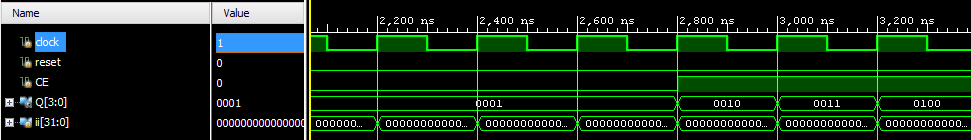
The last thing to consider when writing the code for your clock is the decimal point signal. This should only be shown on the first and third anode, and remember they are active low. Implementing the display itself is simple, as you can use the module used in previous labs.

|  |
| --- |
| Results |

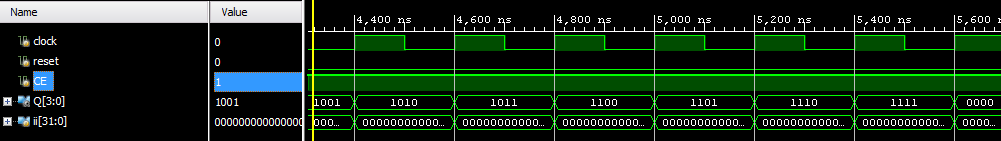
Results open with an effective statement of overall findings presents visuals clearly and accurately, presents findings clearly and with sufficient support. You MUST



Test bench showing Reset only resetting at rising clock edge



Test bench showing CE being disabled, stopping the count



Test bench showing normal behavior of counter, including cycling back around to 0000

|  |
| --- |
| Discussion |

Discussion opens with an effective statement on the goals of the lab, backs up the statement concerning appropriate findings, provides a sufficient and logical explanation for the statement, addresses other issues pertinent to the lab.

The goal of the lab was to create a digital clock. The results above show (using a manually coded clock) how the count enable and reset signal behave. Those signals were relatively easy to implement, however getting the appropriate behavior on the counting was the most difficult. Ensuring that the signal was sent to the next bit without causing the other bits to start randomly counting was solved by putting the reset/count enable on the 9 count (5 for the minutes), and setting it to 0 at all other counts.

The decimal points on anodes 3 and 1 were also slightly difficult, originally I was trying to code two different ones but then found a logic expression that showed both with one expression.

|  |
| --- |
| Conclusion |

Conclusion convincingly describes what has been learned in the lab.

At the end of this lab I have learned a bit about sequential logic, especially in always statements. In an always block that uses a posedge or negedge of a clock, all statements should be non-blocking (<=). I also learned more about utilizing the four anodes of the 7 segment display, as well as manipulating the frequency of the clock.

|  |
| --- |
| References |

Include your references here

Lab 6 Description