ECE 272 Lab 5

System Verilog

Hyunjae Kim

08/04/2021

Matthew Shuman

1. Introduction

This should be at least a paragraph, if not more, about the purpose of the lab, the software/hardware used, definitions of terminology and any knowledge needed to understand what is happening in the lab. This paragraph should give your reader enough information to understand and follow your process through this report (from your design work to your results and so on)

In this lab section, the general purpose is to use System Verilog to build digital blocks, and then combine them to create a clock on FPGA, DE10-Lite, by using Quartus software, and verify the design with hardware output and ModelSim software simulation.

To build a clock on FPGA, digital blocks are required because FPGA clock is consisted of combination of required digital blocks. In required digital blocks, there are counter, comparator, synchronizer, parser, and seven segment displays. The entire digital blocks will be designed by using System Verilog HDL unlike previous lab sections. Moreover, understanding the clock input is necessary. The reason is that the 10MHz clock input will be divided into 1Hz by using multi-bit counter.

The clock built on FPGA indicates seconds to hours. The seconds and minutes will be demonstrated up to 59 on seven segment displays, and the hours will be demonstrated up to 23 on seven segment displays. Moreover, seconds, minutes, or hours will be reset when the time is over 59 seconds, 59minutes, or 23 hours.

To indicate precise time on the clock, the clock input of FPGA clock has to be divided by multi-bit counter. By using multi-bit counter, the clock input will be divided to approximately 1Hz, and then seven segment displays will indicate the time. However, using only multi-bit counter to divide the clock input has some limit to demonstrate precise time. Thus, there will be some addition on clock division to make more precise clock.

After making the FPGA clock, the clock will be verified by checking the hardware output on DE10-Lite. If the hardware outputs seconds to hours, then the hardware verification is successful. Moreover, there can be some errors in the implementation which is not notified by hardware so that it will be verified again by using ModelSim simulation software. If the simulation indicates the expected outputs, then the design of FPGA clock is completely successful.

2. Design

1) Counter
   The counter counts by 1 when the clock input is rising edge, and then outputs N-bit output q. If the reset is 1, then the output is zero because the reset is active high. Moreover, if the reset is 0, then the N-bit output will add by 1 as the clock input flows. (Appendix A)

2) Comparator
   The comparator compares the input value and the parameter M. The output logic is 1 when the N-bit input value is greater than the parameter M, and if the N-bit input value is equal or less than the parameter M, then the output logic is 0. (Appendix B)

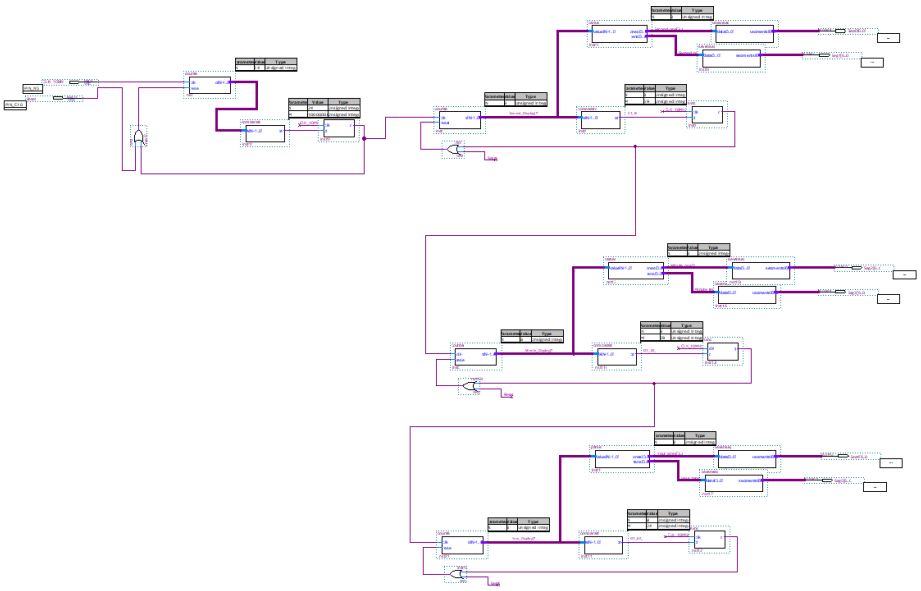
3) Synchronizer
   The synchronizer synchronizes the input value to input clock. To be specific, the input of synchronizer acts as clock input. Moreover, if the input has irregular matching point with the clock input, then it is complicated to perform the expected result. The synchronizer synchronizes the input which acts as the clock to the clock input in order to perform the expected result. (Appendix C)

4) Parser
   The parser splits the binary input value by ones-place and tens-place in decimal, and then sends those value to seven segment displays to each display driver in order to represent the input value in decimals. (Appendix D)

5) Seven Segment Display
   Seven segment displays, which are active low, demonstrate the divided clock input in decimal values. Unlike the past lab section which handled seven segment displays, the displays only demonstrate 0 to 9 because of showing the time in seconds, minutes and hours. On FPGA, there are 6 seven segment displays so that it is available to indicate seconds, minutes, hours. (Appendix E)

6) FPGA Clock

I. Schematic Design

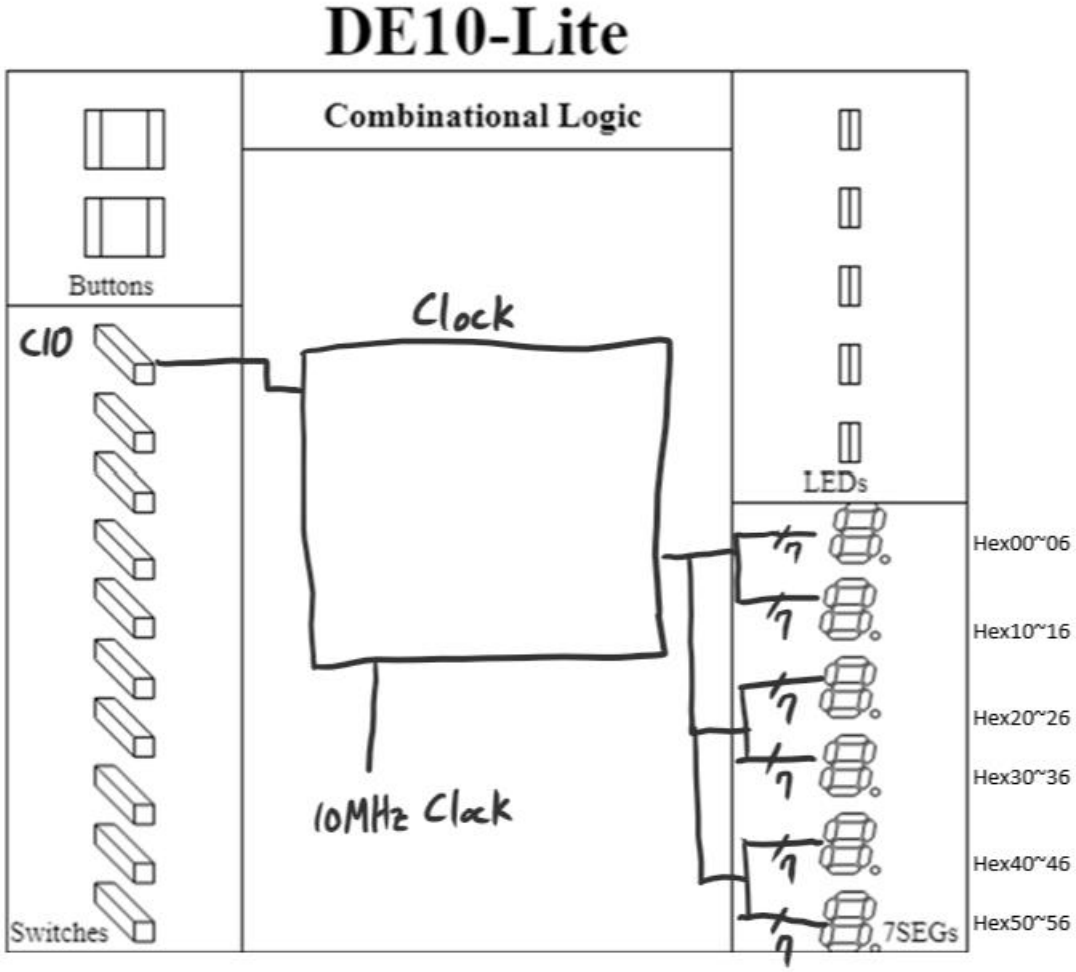


The 10MHz clock input is connected to 24-bit counter, and the output of 24-bit counter is connected to the input of comparator which the value of M is 10million. Then the output of comparator is connected to synchronizer, and the output of synchronizer is connected to OR gate with reset in order to determine the 24-bit counter. By connecting 10MHz clock input, it is much more precise to divide the clock into 1Hz.

The output of synchronizer is connected to 8-bit counter, and then the output of 8-bit counter is connected to parser and comparator to display seconds. The parser splits the input value into ones and tens in decimal, and the combination of comparator and synchronizer determines to reset the 8-bit counter. Then, the output of synchronizer is connected to another 8-bit counter. This process is repeated for 2 times more to display minutes and seconds.

The reason of connecting the output of synchronizer to the input of the counter is that the output of synchronizer acts as divided clock input, and as this is repeated, the clock division goes much more precise.

II. Block Design



The clock is connected to reset, pin C10, and 10MHz clock input. If the reset is 1, then the seven segment displays will not show anything. In the block design, the 10MHz clock input is divided so that seconds are displayed on Hex00~06 and Hex10~16, minutes are displayed on Hex 20~26 and Hex 30~36, and hours are displayed on Hex 40~46 and Hex 50~56.

III. Pin Assignment

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
CLK_10MHz	Gate input	Input	1	ADC_CLK_10	N5
Reset	Gate input	Input	1	Switch 0	C10
Seg0[0]	Gate output	Output	1	Hex00	C14
Seg0[1]	Gate output	Output	1	Hex01	E15
Seg0[2]	Gate output	Output	1	Hex02	C15
Seg0[3]	Gate output	Output	1	Hex03	C16
Seg0[4]	Gate output	Output	1	Hex04	E16
Seg0[5]	Gate output	Output	1	Hex05	D17
Seg0[6]	Gate output	Output	1	Hex06	C17
Seg1[0]	Gate output	Output	1	Hex10	C18
Seg1[1]	Gate output	Output	1	Hex11	D18
Seg1[2]	Gate output	Output	1	Hex12	E18
Seg1[3]	Gate output	Output	1	Hex13	B16
Seg1[4]	Gate output	Output	1	Hex14	A17
Seg1[5]	Gate output	Output	1	Hex15	A18
Seg1[6]	Gate output	Output	1	Hex16	B17
Seg2[0]	Gate output	Output	1	Hex20	B20
Seg2[1]	Gate output	Output	1	Hex21	A20
Seg2[2]	Gate output	Output	1	Hex22	B19
Seg2[3]	Gate output	Output	1	Hex23	A21
Seg2[4]	Gate output	Output	1	Hex24	B21
Seg2[5]	Gate output	Output	1	Hex25	C22

Seg2[6]	Gate output	Output	1	Hex26	B22
Seg3[0]	Gate output	Output	1	Hex30	F21
Seg3[1]	Gate output	Output	1	Hex31	E22
Seg3[2]	Gate output	Output	1	Hex32	E21
Seg3[3]	Gate output	Output	1	Hex33	C19
Seg3[4]	Gate output	Output	1	Hex34	C20
Seg3[5]	Gate output	Output	1	Hex35	D19
Seg3[6]	Gate output	Output	1	Hex36	E17
Seg4[0]	Gate output	Output	1	Hex40	F18
Seg4[1]	Gate output	Output	1	Hex41	E20
Seg4[2]	Gate output	Output	1	Hex42	E19
Seg4[3]	Gate output	Output	1	Hex43	J18
Seg4[4]	Gate output	Output	1	Hex44	H19
Seg4[5]	Gate output	Output	1	Hex45	F19
Seg4[6]	Gate output	Output	1	Hex46	F20
Seg5[0]	Gate output	Output	1	Hex50	J20
Seg5[1]	Gate output	Output	1	Hex51	K20
Seg5[2]	Gate output	Output	1	Hex52	L18
Seg5[3]	Gate output	Output	1	Hex53	N18
Seg5[4]	Gate output	Output	1	Hex54	M20
Seg5[5]	Gate output	Output	1	Hex55	N19
Seg5[6]	Gate output	Output	1	Hex56	N20

3. Results

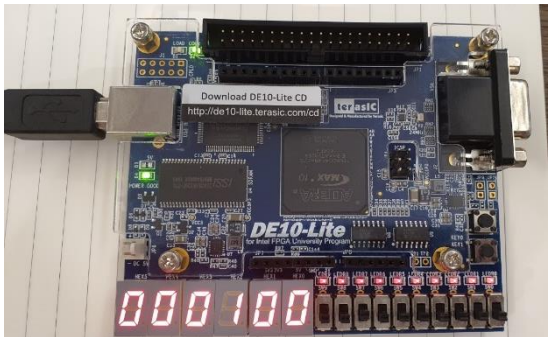
1) Hardware Results

Reset	10MHz Clock	Expected Result	Actual Result
0	Connected to 10MHz clock	FPGA clock flows	FPGA clock flows
1	Connected to 10MHz clock	FPGA clock does not flow	FPGA clock does not flow

When the reset is 1, the FPGA clock does not flow, and when the reset is 0, the FPGA clock flows demonstrating seconds, minutes, and hours.

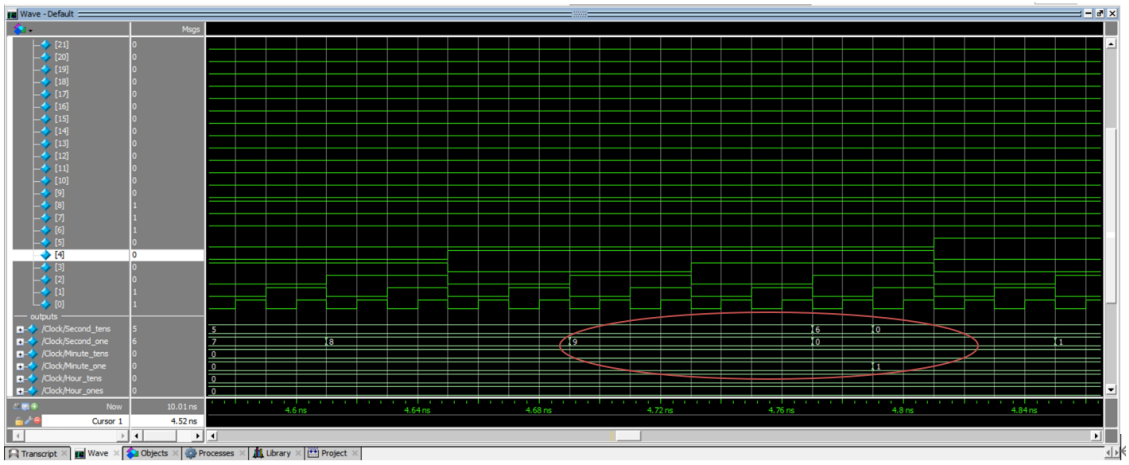
- When reset is 1

- When reset is 0



The FPGA clock demonstrates the time as expected.

2) Simulation Result



In the simulation, FPGA clock counts 60 seconds, and then reset the second display, and add 1 to the minute display. Therefore, the simulation acts as expected results.

The verification via both DE10-Lite and ModelSim simulation, was matching with expected results. Therefore, the lab is successful.

4. Experiment Notes

Most of the parts of this lab section went well. The hardest part is how to implement seconds, minutes, and hours to the hardware. At first, I used 40-bit counter to divide the clock, and assigned 23-bit, 30-bit, and 38-bit to seconds, minutes, and hours for each. However, the output was not correct because when the second is goes to 30 seconds, the display of minute goes to 1. Then, I found that the output of synchronizer acts as a clock so that I connected the output of second's synchronizer to the input of minute's counter and so on. Then the output was same as I expected.

After solving this problem, there was another problem. It was displaying the clock more precisely. I compared my FPGA clock to smartphone stopwatch. FPGA clock was about 20% faster than smartphone stopwatch. Professor advised me to use comparator and synchronizer for clock division. I used comparator and synchronizer for clock division, and set the parameter N to 24, and M to 10million. The value of 2^{24} is much larger than 10million, and 10MHz means that there is 10million changes per second. So, setting parameter M to 10million will change the output of synchronizer to 1Hz clock input. Therefore, using this method made my FPGA clock similar to my smartphone stopwatch.

5. Study Questions

How off is your clock?

(how much does it differ from a minute on a stopwatch/phone)

When I compare my FPGA clock to stopwatch, there was no visual difference of both digital devices. However, a minute on a stopwatch and FPGA clock will have difference. The reasons are that the 10MHz clock input on FPGA may not be exactly equal to 10MHz, and there are some delays on FPGA hardware in pico-seconds. Thus, although there is no visual difference between both stopwatch and FPGA clock, there will have difference in pico-seconds between a stopwatch and FPGA clock.

6. Appendix

A) Counter

```
1   module counter # (parameter N = 8)
2                    (input logic clk,
3                     input logic reset,
4                     output logic [N-1: 0] q);
5
6     always_ff@(posedge clk, posedge reset)
7       if(reset) q <= 0;
8       else      q <= q+1;
9   endmodule
10
```

B) Comparator

```
1   module comparator # (parameter N = 8, M = 60)
2                       (input logic [N-1 : 0] a,
3                        output logic gt);
4
5       assign gt = (a > M);
6
7   endmodule
8
```

C) Synchronizer

```
module sync(input logic clk,
            input logic d,
            output logic q);


   logic nl;

   always_ff@(posedge clk)
      begin
         nl <= d; //nonblocking
         q <= nl; //nonblocking
      end
endmodule

```

D) Parser

```
 module parser # (parameter N = 8)
                 (input logic [N-1 : 0] value,
                  output logic [3 : 0] ones, tens);

  assign ones = value % 10;
  assign tens = (value / 10)%10;

 endmodule


```

E) Seven Segment Displays

```
module sevenseg(input logic [3 : 0] data,
                output logic [6 : 0] segments);

  always_comb
    case(data)
       //Note that 7_seg are active low
       //                          gfe_dcba
       0:       segments =  7'b100_0000;
       1:       segments =  7'b111_1001;
       2:       segments =  7'b010_0100;
       3:       segments =  7'b011_0000;
       4:       segments =  7'b001_1001;
       5:       segments =  7'b001_0010;
       6:       segments =  7'b000_0010;
       7:       segments =  7'b111_1000;
       8:       segments =  7'b000_0000;
       9:       segments =  7'b001_1000;
       default: segments =  7'b111_1111;
    endcase
endmodule

```