Digital Design (CS F215) **28-10-2024**

Lab 10: Counters

Lecturer: Harikrishnan N B *Student:*

READ THE FOLLOWING CAREFULLY:

Honour Code for Students: I shall be honest in my efforts and will make my parents proud. Write the oath and sign it on the assignment.
This lab assignment is a take-home exercise, and there are no in-person evaluations or TA supervision required.

The lab evaluation consists of two parts.

1) **Step 1 (Implementation):** Use iverilog to implement your design. Follow the modeling specified in each question, as only solutions using the instructed modeling will be accepted for evaluation.

2) **Step 2 (Submission):** Once you complete the implementation, create a zip file named `CampusID_Lab10.zip` containing all relevant files and upload it to `quanta.bits-goa.ac.in (local quanta)`.

Note: The submission format should be strictly followed: All files must be organized as `CampusID_Lab10.zip > CampusID_Lab10 (folder with same name) > all files.` **Assignment will be graded using Python script and submission in wrong format will result in direct 0 marks.**

`Note: The clock period is 2 seconds (with 50% duty cycle). Initialize the clock to 0 for all the testbenches.`

Welcome, students! You are a scientist working at ISRO, and you have just discovered a new planet named X1b. However, on this strange planet, time flows differently than on Earth! You've been assigned the task of building a digital clock that functions according to the unique time system of X1b:

- 1 minute = 64 seconds
- 1 hour = 32 minutes
- 1 day = 16 hours

Your mission is to design, build, and display this clock using counters and seven-segment displays.

Part 1: Building Counters for Planet X1b Clock

Q1: Design the Counters for Planet X1b Clock

(a) **Seconds Counter:** Create a counter that increments on every clock cycle. After 64 counts, it resets to 0 and sends a tick signal to the minutes counter.

(b) **Minutes Counter:** The minutes counter increments by 1 only when it receives a tick signal from the seconds counter. After 32 counts, it resets to 0 and sends a tick signal to the hours counter.

(c) **Hours Counter:** This counter increments by 1 whenever it receives a tick signal from the minutes counter and seconds counter. After 16 counts, it resets to 0, representing the end of a day.

Implementation: Implement `seconds.v`, `minutes.v`, and `hours.v` using behavioral modeling, and `clock.v` using structural modeling. `clock.v` is the top module which contains instances of other three modules.Use `tb_clock.v` to check correctness of your modules.

Hint: The tick signal plays a vital role in coordinating the operation of the counters. It serves as an indicator that one counter is about to overflow and that the next counter should increment accordingly. Students should consider when to generate the tick signal to ensure that the timekeeping between seconds, minutes, and hours is accurately maintained. Reset is used to initialize clock to 00:00:00.

Part 2: Designing the Seven-Segment Display Register

In this part, you need to create a seven-segment display register that can display the numbers 0 to 9. Below is the truth table for the seven-segment display based on the given encoding:

Table 10.1: Truth Table for Seven-Segment Display

Digit	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**Binary Encoding**
0	0	1	1	1	1	1	1	7'b0111111
1	0	0	0	0	1	1	0	7'b0000110
2	1	0	1	1	0	1	1	7'b1011011
3	1	0	0	1	1	1	1	7'b1001111
4	1	1	0	0	1	1	0	7'b1100110
5	1	1	0	1	1	0	1	7'b1101101
6	1	1	1	1	1	0	1	7'b1111101
7	0	0	0	0	1	1	1	7'b0000111
8	1	1	1	1	1	1	1	7'b1111111
9	1	1	0	1	1	1	1	7'b1101111

Explanation of the Table:

- **Columns a to g:** Each column corresponds to a segment on the display.
- **0 or 1:** A value of 1 means the segment is ON, and 0 means it is OFF.
- **Binary Encoding:** This shows the 7-bit encoding corresponding to the state of the segments for each digit (0 to 9).

Q2: Implement the Seven-Segment Display Register

Complete `seven_segment.v` using behavioral modeling to implement a module that maps the 4-bit binary input to the appropriate 7-segment outputs. Follow truth table given above.

Part 3: Connecting the 7-Segment Displays to Show Time (HH:MM:SS)

To display the current time, you will need a total of 6 seven-segment displays: 2 for Seconds, 2 for Minutes, and 2 for Hours.

- **Seconds Counter:** Updates the two seconds displays.
- **Minutes Counter:** Updates the two minutes displays.
- **Hour Counter:** Updates the two hours displays.

Example display format (for reference only):

HH : MM : SS

15 : 31 : 63

Connection: Each pair of displays will be connected to their respective counters. The output of the counters should be sent to the display module to show the current time in the format specified.

Q3: Write code to use 6 seven-segment displays to display hours, minutes, and seconds as per the clock on Planet X1b. Students are required to implement the `display.v` module to handle the display logic.

Part 4: Integrating the Counters with the Display

In the final step, you will integrate the counters with the seven-segment displays.

Q4: Implement **main_module.v**, which connects **clock.v** to **display.v** to complete the digital clock.

After implementing the above, use **tb_main.v** to simulate **main_module.v**. Additionally, there is a **driver.cpp** file provided, which is used to display the output of the testbench in a clock format.

We have provided a **demo.txt** file that will be read by **driver.cpp** to generate a pattern mirroring the digital clock with onlycertain part of clock. You need to generate pattern which mirrors digital clock starting from `00:00:00`.

To direct the testbench output to a text file, use the following command:

```
vvp a.out > input.txt
```

In the **driver.cpp** code, ensure to change the file name in the `readPatterns()` function to read your testbench output stored in input.txt and check its validity.

For Windows users, convert your `input.txt` to UTF-8 encoding using Notepad by selecting `Save As` and choosing UTF-8 in the encoding options, otherwise your input file won't be read by driver.cpp.

Mark Distribution

- **seconds.v** : 5 Marks

- **minutes.v** : 5 Marks
- **hours.v** : 5 Marks
- **display.v** :5 Marks
- **clock.v** : 3 Marks
- **main_module.v** : 4 Marks
- **seven_segment.v** : 3 Marks

Total: 30 Marks

Good luck, scientists! May your clock run perfectly on Planet X1b, helping future explorers track time accurately in the unknown depths of space.