Digital Design (CS F215) **23-09-2024**

Lab 6: Flip Flops

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.
For the lab, you need a dedicated fresh notebook. Keep this notebook dedicated for solving the questions pertaining to lab.

There will be design questions in the lab, the students are supposed to theoretically show the correctness of their design function. Following that, the students are encouraged to use iverilog software to implement the same.

The lab evaluation consists of two parts.

1) Step 1 (Theory): The students are supposed to solve the design question in their dedicated lab notebook. They are supposed to get it checked and validated by TA's/ instructors. (**Deadline for theory evaluation: 10:15 AM**)

2) Step 2 (Lab): Once Step 1 is completed, they can use iverilog to implement their design. Once you have completed the design, inform the TA's/ instructors and get it verified. (**Deadline for lab file submission in local quanta: 10:40 AM**)

3) Step 3: Upload your solution to quanta.bits-goa.ac.in (local quanta). Create a zip file CampusID_Lab6.zip which contains all the files.

`Note: The clock period is 10 seconds. Initialize the clock to 0 for all the testbenches.`

All questions must be solved using the modeling mentioned in the question. Only solutions implemented using the mentioned modeling will be accepted for evaluation.

---

1. **Theory(2 Marks):** Draw the block diagram for 1-bit D flip-flop. Clearly label all inputs and outputs. Your diagram should include a posedge clock input (`clk`), an active-low asynchronous reset input (`reset`), a data input (`d`), and an output (`q`). Also, make the flip-flop characteristic table.

2. **Lab(4 Marks):** Complete the module `dff_async.v` in Verilog using a behavioral model to implement a 1-bit D flip-flop.The module should take a posedge clock input (`clk`), an active-low asynchronous reset input (`reset`), and a data input (`d`), producing an output q. After completing the module, write the corresponding testbench (`testbench_dff_async.v`) and use `table 6.1` to verify the functionality of your D flip-flop.

3. **Theory(3 marks):** Design an 8-bit D flip-flop using 1-bit D flip-flop. Clearly label all inputs and outputs. Your diagram should include a posedge clock input (`clk`), an active-low asynchronous reset input (`reset`), an 8-bit data input (`d[7:0]`), and an 8-bit output (`q[7:0]`)

Table 6.1: Testcases for the 1-bit D flip-flop.

Time	**reset**	**d**
0	1	0
10	1	1
20	1	0
30	1	1
37	0	1
43	1	1
53	1	0
60	1	1
70	1	1

4. **Lab(7 marks):** Complete the module `dff_async_8bit.v` in Verilog using a structural model to implement the above circuit. The module takes posedge `clk`, active-low `reset` and `d[7:0]` as the input and gives `out[7:0]` as the output. Complete the corresponding testbench (`testbench_dff_8bit.v`). Verify the correctness of the module for the testcases provided in below Table 6.2. Testcases should be in the order given below.

Table 6.2: Testcases for the 8-bit D Flip-flop.

Time	**d[7:0]**	**reset**
0	0000 0000	1
10	1010 1010	1
20	0110 0110	1
30	1111 0000	1
37	1111 0000	0
43	1111 0000	1
53	0000 1111	1
60	1111 1111	1
70	1111 1111	1

5. **Theory(6 marks):** Design a special memory using an active-low reset signal. When the reset is asserted low, the memory is initialized to 0000 0000. When the reset is asserted high,and if the userinput signal is set to 0, the memory will load a default value of 1111 1111 and if the userinput signal is set to 1, the memory will load an 8-bit input from the user.You can assume that an 8-bit 2:1 multiplexer and basic gates are given.

6. **Lab(8 marks):** Complete the memory.v module in Verilog using structural modeling. The module takes a posedge clock input (clk), an active-low asynchronous reset (reset), an 8-bit user input (user), and a control signal (userinput). You will use an 8-bit 2:1 multiplexer (mux2_1.v) to select between the user input and the default value, and an 8-bit D flip-flop with active-low asynchronous reset to store the selected output.

Table 6.3: Testcases for mux2_1.

Time	D0[7:0]	D1[7:0]	S
0	0000 0000	1111 1111	0
10	1010 1010	0101 0101	0
20	1010 1010	0101 0101	1
30	0001 1101	1110 0010	0
40	0001 1101	1110 0010	1
50	1100 1100	0011 0011	0
60	1100 1100	0011 0011	1
70	1111 1111	0000 0000	0
80	1111 1111	0000 0000	1

Table 6.4: Testcases for memory.

Time	user[7:0]	reset	userinput
0	1100 0101	1	1
10	1111 0101	1	1
12	1111 0101	1	0
18	1111 0101	0	0
20	1100 1100	0	0
24	1100 1100	0	1
28	1100 1100	1	1
36	1100 1100	1	0
40	1111 0000	1	1
50	0001 0001	1	1
60	0001 0001	1	1

Instructions for Lab 6

NOTE: Complete all the modules using the modeling mentioned in the question.
NOTE: Make sure to **not modify the monitor line in testbench**, and keep the **module name** and **input/output variable names unchanged**.
NOTE: Make sure you **don't** change the order of the test cases.

1. To test you code using testbench `testbench_name.v` **How to test:**

   ```
   iverilog testbench_name.v
   vvp a.out
   gtkwave (file with extension .vcd created)
   ```

2. Complete the `dff_async.v` module and `testbench_dff_async.v`.

   How to test:

   ```
   python3 autograder.py
   ```

3. Complete the `dff_8bit.v` module and `testbench_dff_8bit.v`.

   How to test:

   ```
   python3 autograder.py
   ```

4. Complete the `memory.v` module using the `mux2_1.v` module.
   Also complete `testbench_memory.v`

   How to test:

   ```
   python3 autograder.py
   ```

NOTE: After completing all the verilog files, create a **zip file** named **ID_Lab6**. Make sure the zip file **does not have multiple folders inside it**.