EC311 Introduction to Logic Design
Spring 2022
Lab 2: Flip Flops and MUXs

Goals

- Understanding edge triggered devices
- Getting familiar with different types of flip-flops and clock-dividing
- Design and implementation on the FPGA of simple sequential circuits

Abstract

In this lab, you will use several new Verilog constructs. The first one is the concept of "POSEDGE" and "NEGEDGE" which are used to trigger devices when the clock rises or falls. This trigger occurs "instantaneously", not sequentially, for all the devices connected to the same clock. You will start with implementing a simple D-Flip-Flop (D-FF) and add on top of it. One thing you will add is a reset that operates both synchronously with the clock and another reset that is asynchronous with respect to the clock. The statements placed within your code will determine whether the reset is synchronous or asynchronous. For this purpose, you will use the construct **"always"**, with a sensitivity list, which means that when certain "watched" signals in the sensitivity list change, a response is triggered with a small delay. In this lab, you will also use **"case"**, which enables you to choose between options according to the input signal. For example, using "case" is a good way to implement multiplexers (MUX), even though constructs like (**?:**) or **if**/**else** can be used similarly. In this lab, you will use a D-FF to make a T-FF and a JK-FF, thus reinforcing the concepts you learned in class.

Tasks

1. **Flip-flops**

In this part, you will implement three different types of FFs with two different reset types. You must show your results using simulation. You must use **behavioral** Verilog for the FFs.

Steps:

1. Build a positive edge triggered D-FF.
2. Add a **synchronous** reset as an input to the D-FF.
3. Make a new module and reuse your old code to add an **asynchronous** reset to the D-FF.
4. Implement a positive edge triggered T-FF using the D-FF that you have implemented in previous steps.
5. Add a **synchronous** reset as an input to the T-FF.
6. Make a new module and reuse your old code to add an **asynchronous** reset to the T-FF.
7. Use the D-FF to construct a JK-FF.
    a. Use the DFF with **synchronous** reset to build a JK-FF with **synchronous** reset.
    b. Use the DFF with **asynchronous** reset to build a JK-FF with **asynchronous** reset.
8. Instantiate two versions of JK-FFs with synchronous and asynchronous resets in your top module.
    a. Two JK-FF modules will share the same inputs `J`, `K`, `reset`, and `clock`.
    b. The final outputs of the top module will be the output `Q_syn` from JK-FF with

synchronous reset and `Q_asyn` from JK-FF with asynchronous reset.

For each step, build a testbench that shows your modules work. Test your modules exhaustively.

2. Clock Divider

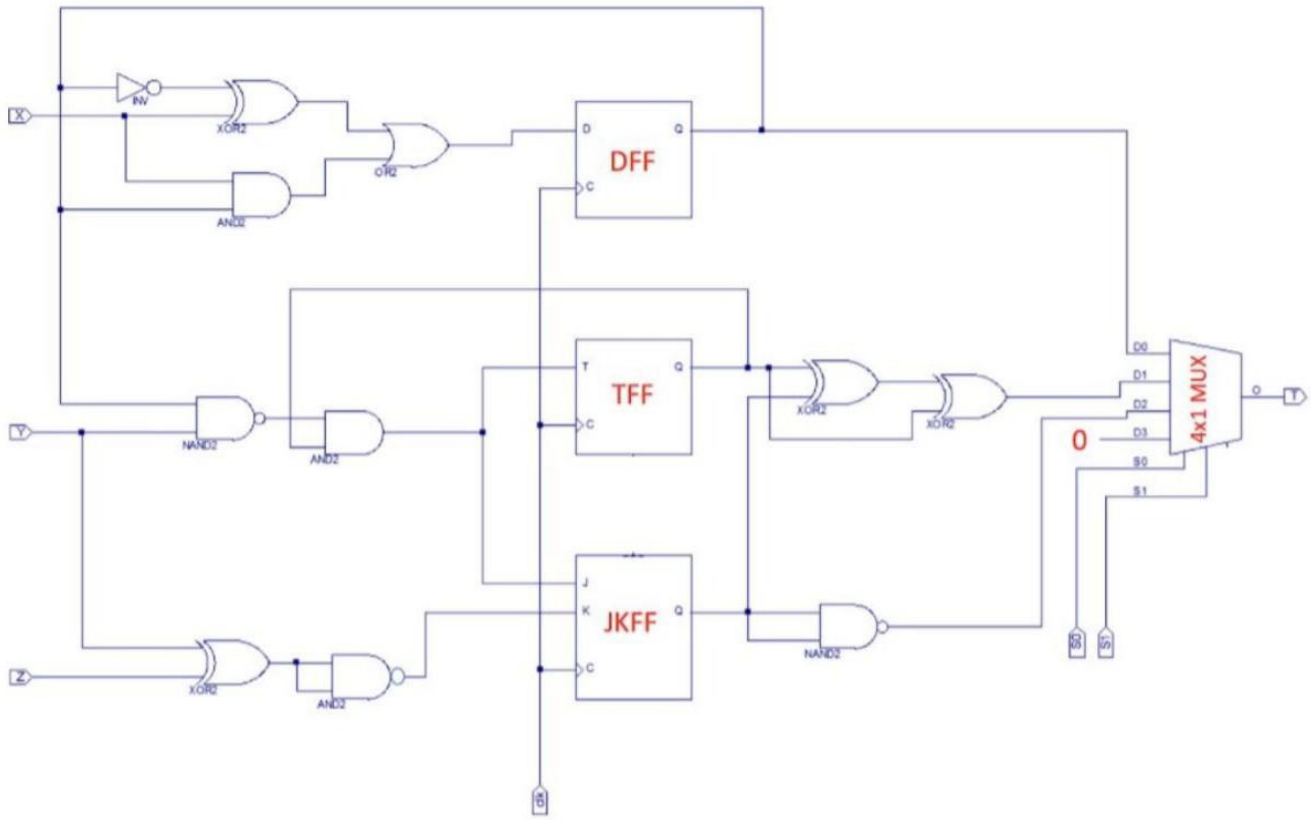
Use the D-FF and the clock divider provided to you and divide the system clock by 10 to create a 10x slower clock and use it as the clock of the D-FF. Run a testbench and verify the D-FF changes output at posedge of the slower clock.

The reason you want that clock to be slow is that when you move the switches (in case you implement the design on FPGA), you can observe the DFF output. We provided you a clock divider Verilog file (clk_divider.v). You need to understand the implementation of this file and make changes to satisfy the required clock.

3. Implement the Circuit

Implement the following circuit using the modules that you have implemented in Task 1. For FFs, you should use one single synchronous reset. You have 5 input ports, i.e. `X`, `Y`, `Z`, `S0`, `S1` and one output port `T`.

Circuit



4. FPGA Implementation

Push the design of your T-FF on the FPGA and show the output. The top-level model will have input signal `T` (as switches), `RESET` (as button), and `CLOCK` (on LED) and output `Q` from TFF (on LED). Demonstrate to the TAs. Use a big clock divider number to slow down the clock to 1 Hz so that you can observe the output. To do this, count up or down using the main system clock. As an example, you could count from 1 to $40x10^6$ (i.e. if the internal clock of the FPGA is 40 MHz) and use the end marker as your 1 Hz clock. To eliminate (sidestep) the issue of a noisy button, hold the button for more than a second when you want to reset.

Deliverables:

a. The verilog module files (.v) for the D-FF, T-FF and JK-FF with both synchronous and asynchronous reset and their testbenches.
b. The verilog module file (.v) for the circuit in Task 3 and its testbench.
c. The constraints file (.xdc) of the T-FF on FPGA.
d. Zip all the verilog files in a compressed folder when you upload to blackboard.

Important: Make sure your code does not have latches.