Middle East Technical University

CENG 232

Logic Design

Spring '2016

Lab - 3

Part 1 Due date: Sunday, April 10, 2016, 23:55hrs
Part 2 Due date: Friday, April 15, 2016, 23:55hrs
No late submissions

1 Introduction

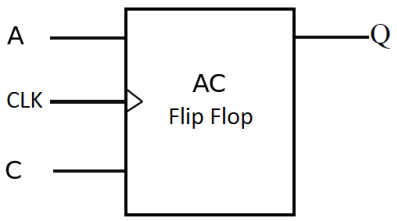
This assignment aims to make you familiar with Verilog language, related software tools, and the FPGA boards. There are two parts in this assignment. The first part is a Verilog simulation of an imaginary flip-flop design, which you are required to implement and test on your own. The second part consists of simulation and implementation (on FPGA) of LogicOLeague system.

2 Part 1: AC Flip Flop (Individual Work)

This part of the lab will be performed and submitted individually.
You are given a specification of a new type of flip-flop, and a new chip that uses the flip flop. Your task is to implement these in Verilog, and prove that they work according to their specifications by simulation.

2.1

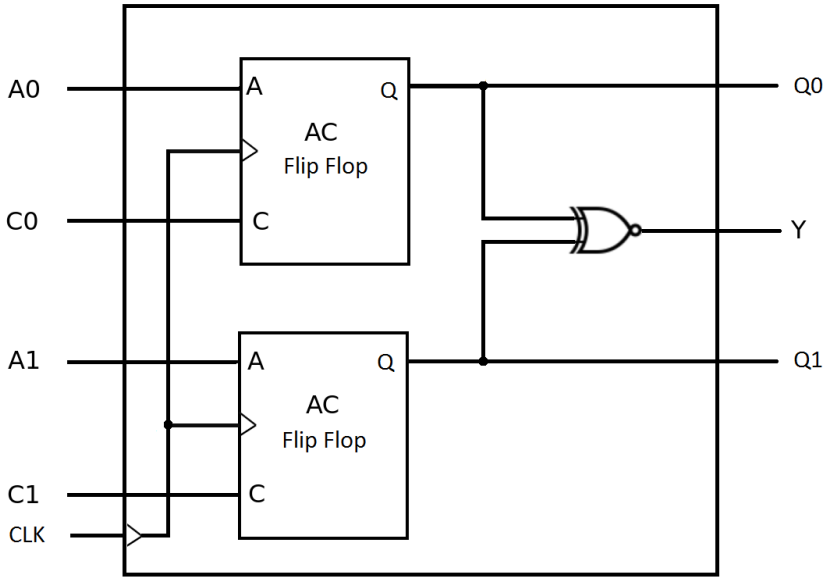
Implement the following AC flip-flop in Verilog with respect to the provided truth table. An AC flip-flop has 4 operations when inputs A and C are: 00 and 11 (set to 1), 01 (complement), 10 (no change). Please note that the AC Flip-Flop changes its state only at rising clock edges.



A	**C**	**Q**	$\mathbf{Q}_{next}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2.2

Implement the following chip that contains two AC flip flops which has output Y.



Use the following module definitions for the modules:

```
module acflipflop(input a, input c, input clk, output reg q)
module ic3031(input a0, input c0, input a1, input c1, input clk, output q0, output
q1, output y)
```

Simulation

A sample testbench for **acflipflop** module will be provided to you. It is your responsibility to extend the testbench, and also to write a testbench for **ic3031** module.

Deliverables

- Implement both modules in a single Verilog file: *lab3_1.v*. Do NOT submit your testbenches. You can share your testbenches on the newsgroup.
- Submit the file through the COW system before the given deadline. **April 10, 2016, 23:55hrs.**
- This part is supposed to be done individually, not with your partner. Any kind of cheating is not allowed.

3 Part 2: LogicOLeague System (Teamwork)

This part of the lab will be performed and submitted with your group partner. Only submit a single copy per group.

3.1 Problem Definition

Chief directors of Ceng Kingdom Football Federation (CKFF) decide to apply an ingenious system called ***LogicOLeague***, to provide higher security standards in football stadiums! ***LogicOLeague*** name is inspired from the excessive logic that the system offers to the football community! In this system, each football fan should have a ***LogicOTicket*** (which is actually the name given to a match ticket in ***LogicOLeague*** system) in order to enter a stadium to watch a football match. Each stadium has 2 gates which are named as **home** and **away**. The system will firstly be tested in only one stadium before put into practice in the other stadiums. The specifications of the system are as follows:

Each ***LogicOTicket*** has a ticket ID printed on it. A ticket ID consists of a 5-bit number. Each ticket ID keeps the information whether a fan is allowed to enter/leave the stadium either from **home** or **away** gate. The stadium has two tribunes: **home** and **away**. If a fan enters from **home** gate, he/she watches the match from **home** tribune. Otherwise, he/she watches the match from **away** tribune. The system has no memory so there may be multiple tickets having the same ticket ID. Your aim is to design the system summarized above having the following specifications:

1. Fans can either enter or leave the stadium with their ***LogicOTicket***.
2. The current number of fans who entered the stadium from any gate is recorded for each tribune seperately. If a fan enters/leaves the stadium, the current number of fans in the corresponding tribune should be updated.
3. The entry of fans to the stadium are arranged as follows:
   - **a)** If the ticket ID includes more 1s than 0s, then the fan should enter the stadium from **home** gate.
   - **b)** If the ticket ID includes more 0s than 1s, then the fan should enter the stadium from **away** gate.
4. If a fan owns a ticket including more 1s than 0s in its ticket ID and he/she wants to enter from **away** gate, he/she is not allowed to enter the stadium. If a fan owns a ticket including more 0s than 1s in its ticket ID and he/she wants to enter from **home** gate, he/she is not allowed to enter the stadium.
5. The system will give a warning if the case described in item 4 occurs.
6. Initially, the tribunes are empty.
7. It is not possible to leave an empty tribune (This case will not be tested).
8. The ticket ID will not be checked whenever a fan leaves the stadium.
9. The number of fans in the tribunes will be shown in hexadecimal form. The number of fans in any tribune exceeding 15 will be ignored and this case will not be tested.

3.2 Sample Input/Output

Current State	CLK	Next State
10110, 0, 1, 0, 0, 0	↑	1, 0, 0
11101, 0, 1, C, 5, 0	↑	D, 5, 0
01010, 1, 1, 4, E, 0	↑	4, F, 0
xxxxx, 0, 0, 9, F, 0	↑	8, F, 0
xxxxx, 1, 0, 3, 1, 0	↑	3, 0, 0
11001, 1, 1, 5, F, 0	↑	5, F, 1

• The values in ***Current State*** column, which are seperated by "**,**" are defined as: **ticketID**, **gate**, **mode**, **numOfFanInH**, **numOfFanInA**, **gateWar**, respectively.
• The values in ***Next State*** column, which are seperated by "**,**" are defined as: **numOfFanInH**, **numOfFanInA**, **gateWar**, respectively.
• "**x**" denotes "don't care" value.

3.3 Input/Output Specifications

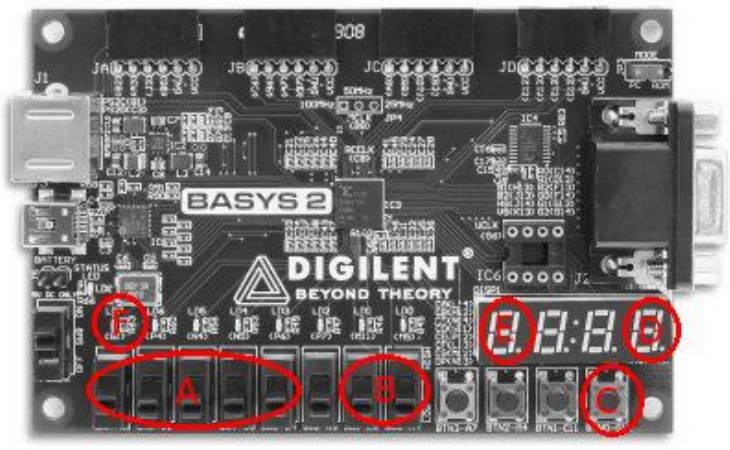
Name	Type	Size
ticketID	Input	5 bits
Clock (CLK)	Input	1 bit
gate	Input	1 bit
mode	Input	1 bit
numOfFanInH	Output	8 bits
numOfFanInA	Output	8 bits
gateWar	Output	1 bit

• **ticketID** represents 5-bit code.
• **CLK** is the clock input for the module.
• **gate** is used for the selection of the gate.
$gate = 0 \Rightarrow$ Home gate
$gate = 1 \Rightarrow$ Away gate
• **mode** is used to indicate whether the fan enters or leaves the stadium.
$mode = 0 \Rightarrow$ Leave
$mode = 1 \Rightarrow$ Enter
• **numOfFanInH**, **numOfFanInA** shows the number of current fans in home and away tribunes, respectively.
• **gateWar** shows whether the fan wants to enter the correct or wrong gate.
$gateWar = 0 \Rightarrow$ correct gate.
$gateWar = 1 \Rightarrow$ wrong gate.

3.4 FPGA Implementation

You will be provided with a Board232.v file (and a ready-to-use Xilinx project), which will bind inputs and outputs of the FPGA board with your Verilog module. You are required to test your Verilog module on the FPGA boards. After the submission date, you will make a demo to course assistants.

Name	FPGA Board	Description
ticketID	SW7,SW6,SW5,SW4,SW3	Left-most 5 switches (A)
Clock (CLK)	BTN0	Right-most button (C)
gate	SW0	Right-most switch (B)
mode	SW1	The switch next to SW0 (B)
numOfFanInH	7-segment display	Right-most 7-segment display (D)
numOfFanInA	7-segment display	Left-most 7-segment display (E)
gateWar	LD7	Left-most led (F)



3.5 Deliverables

- Implement your module in a single Verilog file: *lab3_2.v*. Do NOT submit your testbenches.You may share your testbenches on the newsgroup.
- Submit the file through the COW system before the given deadline. **April 15, 2016, 23:55hrs.**
- This part is supposed to be done with your group partner. Make sure both of you take roles in implementation of the project. Any kind of inter-group cheating is not allowed.
- Use the newsgroup metu.ceng.course.232 for any questions regarding the homework.
- You will make a demo with the FPGA board the next week after the submissions (in your lab session hours). The exact dates and place will be announced later.