Middle East Technical University - Department of Computer Engineering

CENG 232

Logic Design

Spring '2023-2024

Lab 3

Part 1 & Part2 Due Date: 10 May 2024, Friday, 23:55
No late submissions

1 Introduction

This assignment aims to make you familiar with Verilog language and the related software tools. There are two parts in this assignment. The first part is a Verilog simulation of an imaginary flip-flop design. The second part consists of the implementation of a Vending Machine System.

2 Part 1: Warm-up (50 pts)

You are given a specification of a new type of flip-flop, and a new chip that uses this flip-flop. Your task is to implement these in Verilog, and prove that they work according to their specifications by using testbenches.

2.1 AC Flip-Flop Module

Implement the following AC flip-flop in Verilog concerning the provided characteristic table in Table 1. The 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**. **Initially, the output of the flip-flop should be set to one**. set to 1 complement no change set to 0)

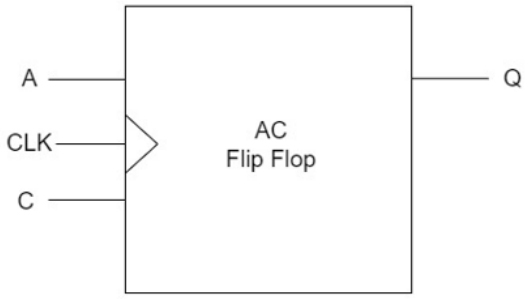

Figure 1: AC Flip-flop diagram

A	C	Q	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

Table 1: AC Flip-flop characteristic table

2.2 ic1406 Module

Implement ic1406 chip given in Figure 2 that contains two AC flip-flops, with A_0, A_1, A_2, and clk as inputs; and Q_0, Q_1 and Z as outputs. Please note that A_0', A_1' and A_2' are the complements of A_0, A_1 and A_2, respectively. Use the following module definitions:

```
module bh(input A, input C, input clk, output reg Q)
module ic1406(input A0, input A1, input A2, input clk, output Q0, output Q1, output Z)
```

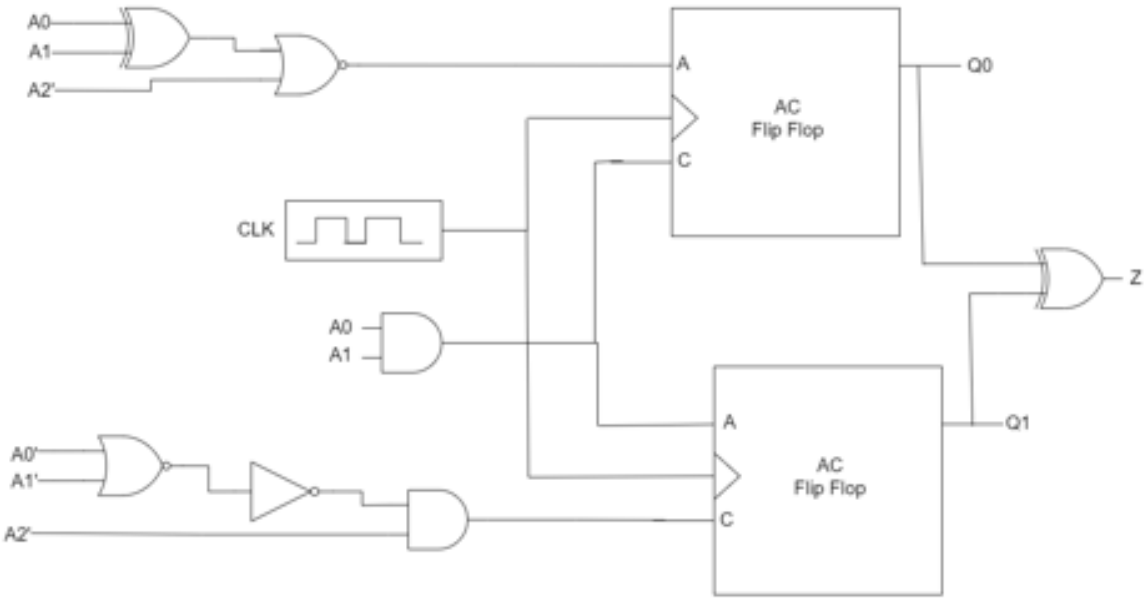

Figure 2: ic1406 module, inputs and outputs.

2.3 Simulation

A sample testbench for AC flip-flop module will be provided to you. You may want to extend the testbench, and also write a testbench for ic1406 module to test different scenarios.

2.4 Deliverables

- Implement both modules in a single Verilog file. Upload only **lab3_1.v** file to ODTUClass system. Do **NOT** submit your testbenches. You can share your testbenches on ODTUClass discussion page.
- Submit the file through the ODTUClass system before the deadline.
- This is individual work, any kind of cheating is not allowed.
- Implementations will be tested using custom test benches in a block-box fashion. We will use **Xilinx** and **iverilog** to conduct these tests. Please make sure that your code compiles over these tools.
- for iVerilog compilation please use **-g2005** flag, it is necessary to align the iVerilog tool with the Verilog standard that Xilinx uses.

3 Part 2: Seating System

In contemporary workspaces, effectively managing resources is essential for enhancing both productivity and comfort. The Seating System is an innovative approach to managing seating allocation and availability within an office setting. Utilizing state-of-the-art technology alongside smart monitoring capabilities, the system facilitates a smooth experience for users to locate and reserve seating options that meet their specific requirements.

3.1 System Overview

You are tasked with implementing a seating system, which is composed of three distinct areas:

1. **Loud Area:** Designed for dynamic interactions and collaboration, offering **15 seats without power plugs.**
2. **Quiet Area:** Intended for focused work, **with 15 seats without plugs and 10 seats with plugs.**
3. **Individual Area:** Tailored for solitary work, providing **20 seats**, all equipped **with power plugs**.

3.2 Operational Overview

- Each **area category** is identified by a unique **2-bit code**.
- The system operates with cash, represented as a **6-bit digital input**.
- Following the insertion of cash and area selection, users may choose a seat with or without a plug, depending on availability and area-specific options.
- The system must internally **track the occupancy of seats, updating availability after each reservation.**

3.3 Area Codes and Fees

Each area is assigned a 2-bit code (`selectedArea`) correlating to the entrance fee structure outlined in Table 2:

- **Loud:** 10 units of currency.
- **Quiet:** 20 units of currency.
- **Individual:** 30 units of currency, with the provision for an additional fee for plug usage.

The following table summarizes the seating options and plug availability across the different areas, along with the entrance fee for each area:

Table 2: Seating options, plug availability, and selection bits

Area	Select Bits	Total	With Plugs	Without Plugs	Fee
Loud	00	15	0	15	10 units
Quiet	01	25	10	15	20 units
Individual	11	20	20	0	30 units

3.4 Transactional Mechanics

Financial transactions within the system are subject to the following conditions:

- **Exact Funds in Individual Area:** Reservations in the Individual area require exact currency matching the area's fee. A deviation will prompt a `notExactFund` warning.
- **Funds in Loud and Quiet Areas:** If the cash provided is less than the selected area's fee, raise `insufficientFund` warning.

Successful transactions will trigger a `seatReady` signal, indicate the number of remaining seats (`seatLeft`), and the residual cash post-purchase (`moneyLeft`)

3.4.1 Example Transaction

Consider a user intending to reserve a seat in the Quiet area with a plug, inserting 25 units of currency:

1. User selects the Quiet area (`01`) and indicates a preference for a plug (`1`).
2. System checks if a plug seat is available and if 20 units suffice for entry.
3. Upon confirmation, the system deducts the fee, reserves the seat, and returns the change.
4. Outputs:
    - `seatReady`: Enabled.
    - `moneyLeft`: 5 units.
    - `seatLeft` (seat left in the Quiet area with a plug): Decremented by 1.

This process showcases how users interact with the system to reserve seats based on their specific requirements. .

3.5 System Protocol and Hierarchy of Warnings

The system operates on a protocol that raises warnings in a predefined order and remains idle in the presence of warnings until the next clock cycle. The warning hierarchy is as follows:

Table 3: Hierarchy of System Warnings

Priority	Warning Signal	Description
1	`seatUnavailable`	No more seats available in the selected area.
2	`invalidPlugSeat`	Attempt to reserve a seat with a plug where none exists.
3	`plugSeatUnavailable`	All seats with plugs in the selected area are reserved.
4	`notExactFund`	Inserted money does not match the required fee exactly.
5	`insufficientFund`	Inserted money is less than the required fee.

Note: In the Individual area, requesting a seat without a plug does not raise an error, as using a plug is optional, not mandatory.
Note: In the Quiet Area, however, requesting a seat without a plug when all such seats had been reserved does raise an error: **seatUnavailable**.

3.6 Input/Output Specifications

Name	Type	Size
money	Input	6 bits
Clock (CLK)	Input	1 bit
plugAvailability	Input	1 bit
selectedArea	Input	2 bits
moneyLeft	Output	6 bits
seatLeft	Output	5 bits
invalidPlugSeat	Output	1 bit
plugSeatUnavailable	Output	1 bit
notExactFund	Output	1 bit
insufficientFund	Output	1 bit
seatReady	Output	1 bit
seatUnavailable	Output	1 bit

3.7 Input and Warning Descriptions

All system operations are synchronized to the positive edge of the system clock.

The seating system operates based on the following inputs and warnings:

- **money**: A 6-bit input representing the cash amount uploaded by the user.
- **CLK**: The clock input that synchronizes the module's operations.
- **plugAvailability**: A binary input indicating whether a seat with a power plug is requested.
- **selectedArea**: A 2-bit input code for selecting one of the areas:
  - `00` ⇒ Loud area, offering seats without plugs.
  - `01` ⇒ Quiet area, offering seats with and without plugs.
  - `11` ⇒ Individual area, offering seats with plugs only.
- **invalidPlugSeat**: Raised when a seat with a plug is requested in an area where no such seats exist (e.g., Loud area).
- **plugSeatUnavailable**: Indicates that all seats with plugs in the selected area are already reserved.
- **seatUnavailable**: A warning signal indicating that no more seats are available in the selected area.
- **notExactFund**: Issued for the Individual area if the inserted money does not exactly match the required fee.
- **insufficientFund**: Triggered in the Loud and Quiet areas if the inserted money is less than the required fee.
- **seatReady**: Enabled when a reservation can be successfully processed.

Additionally, **the system handles warnings based on a predetermined priority**, ensuring that the most critical alerts are addressed first. In the event of any warning, the system withholds processing new reservations until the subsequent positive clock edge.

3.8 FPGA Implementation

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

Name	**FPGA Board**	**Description**	
seatUnavailable	led[7]	(A)	Top-most LED
insufficientFund	led[6]	(B)	Second LED from the top
notExactFund	led[5]	(C)	Third LED from the top
invalidPlugSeat	led[4]	(D)	Fourth LED from the top
plugSeatUnavailable	led[3]	(E)	Fifth LED from the top
seatReady	led[2]	(F)	Sixth LED from the top
money	sw[7:3]	(G)	5 left-most switches
selectedArea	sw[2:1]	(H)	Middle switches
plugAvailability	sw[0]	(I)	Right-most switch
moneyLeft		(J)	7-segment display left 2 digits
seatLeft		(K)	7-segment display right 2 digits

Table 4: Module I-O to FPGA Board I-O Mappings

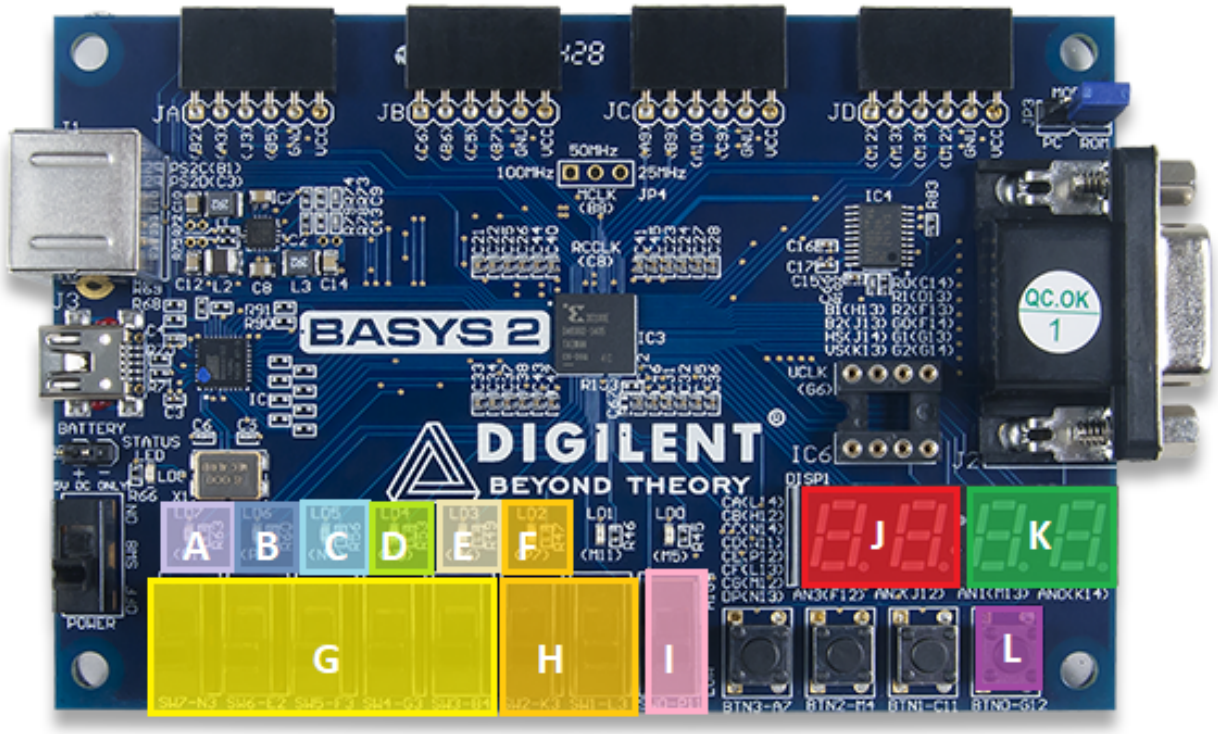

Figure 3: Board figure with labels

4 Template Files

For each part of the assignment, specific template files are provided to assist you in starting your projects. These files are set up with the necessary configuration.

4.1 Part 1: Flip-Flop and IC Implementation

For the first part of the homework, the following template files are provided:

- **lab3_1.v**: This Verilog file defines the modules for the AC flip-flop and the ic1406. You will implement your solution here.
- **part1.xise**: This is the Xilinx ISE project file. It includes the project setup with the desired hierarchy for your modules.
- **test.v**: Includes a test bench for the AC flip-flop.
- **test_ic1406.v**: Contains a test bench for the ic1406 module.

4.2 Part 2: Seating System Implementation

For the second part, the templates are as follows:

- **Board232.ucf**: Defines the board specifics for the FPGA. This file is provided for FPGA testing and requires no modifications.
- **Board232.v**: Defines the FPGA board implementation for testing purposes. This file should not be changed.
- **lab3_2.v**: This is the Verilog module that you need to implement for the seating system.
- **lab3_2.xise**: The Xilinx ISE project file with the correct setup and hierarchy for your FPGA implementation.
- **tester.v**: A simple tester provided for `lab3_2.v`.

5 Loading the Verilog Module onto the FPGA Board

To load the Verilog module onto the FPGA board, please follow the detailed steps below. The accompanying figures provide a visual guide to each step of the process. For full reference, please refer to the **part3_fpga_board_manual.pdf** under **Recitation 2: Verilog Language and Board Usage** on **ODTUCLASS**.

1. Configure the process properties before generating the programming file. Under the "Startup Options" tab, select "JTAG Clock" for the FPGA Start-Up Clock.
2. Generate the programming file by double-clicking "Generate Programming File" under the "Implementation" tab.
3. Connect the FPGA and select the Basys2 board before initializing the scan chain.
4. Ensure the FPGA board is powered on, then click "Browse" to select the .bit file for upload.
5. Choose the "board232.bit" file from your project directory.
6. Press the "Program" button to upload your program to the FPGA board.

After these steps, your Verilog module should be successfully loaded onto the FPGA board. The figures below illustrate these steps:

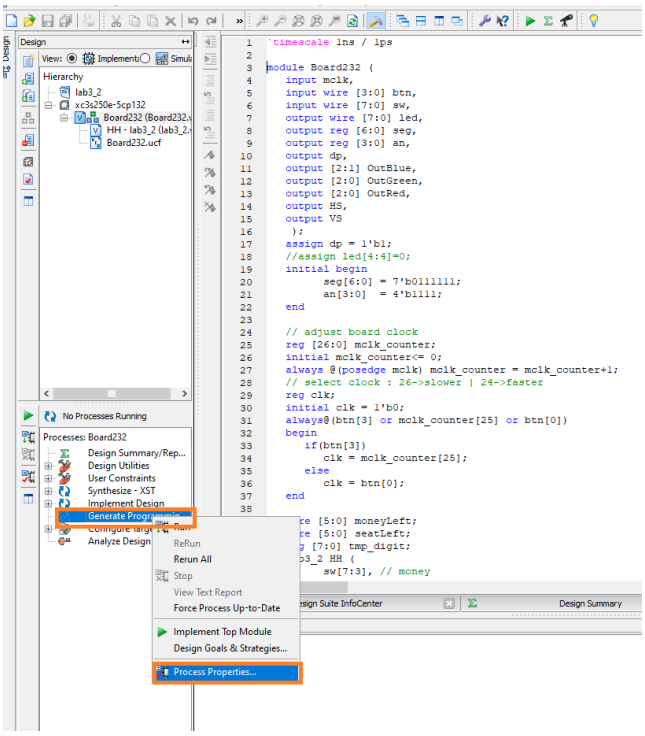

Figure 4: Process Properties configuration.

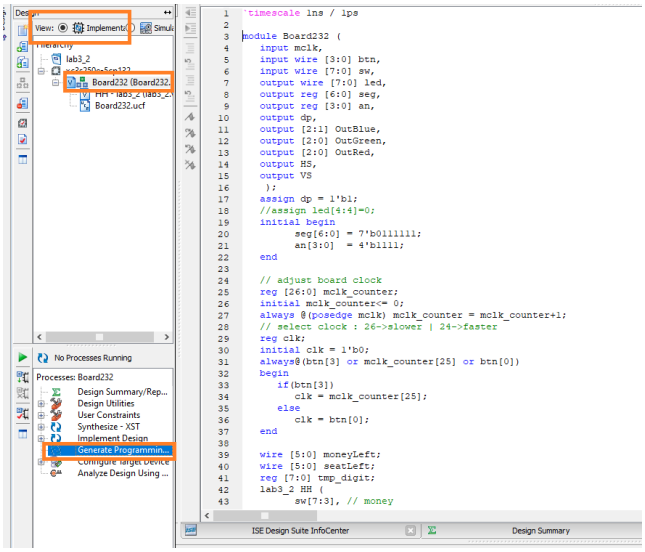

Figure 5: Generating the programming file.

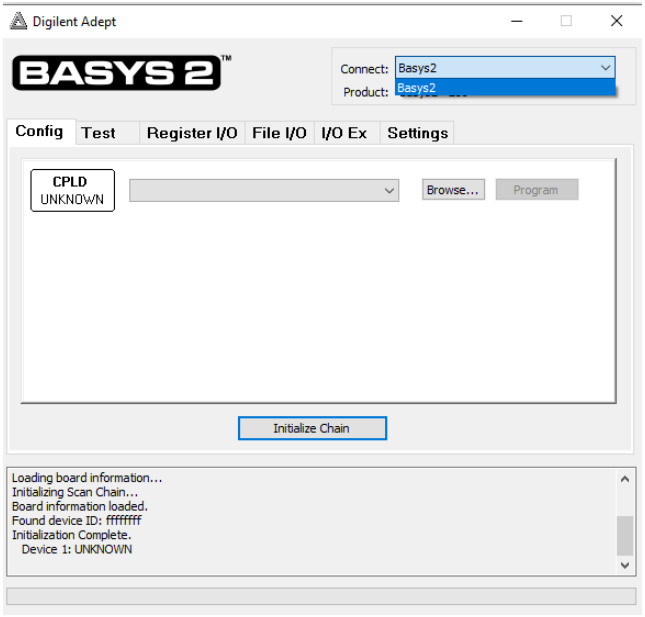

Figure 6: Selecting Basys2 and initializing the scan chain.

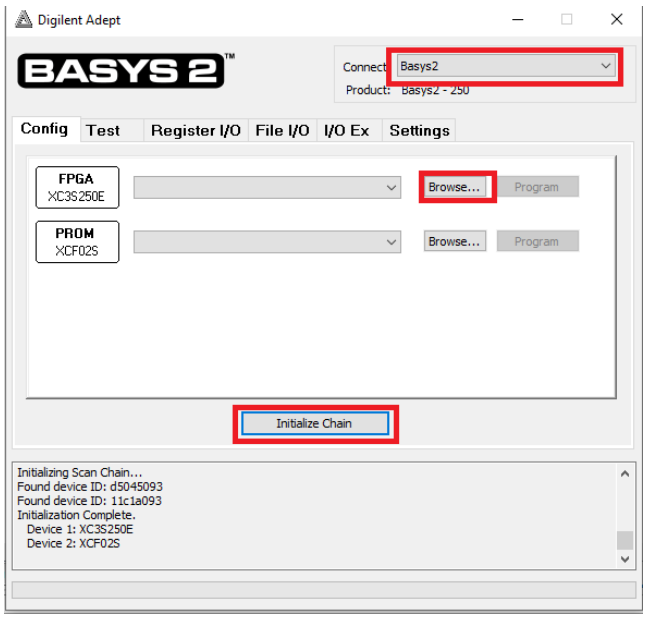

Figure 7: Selecting the .bit file for programming.

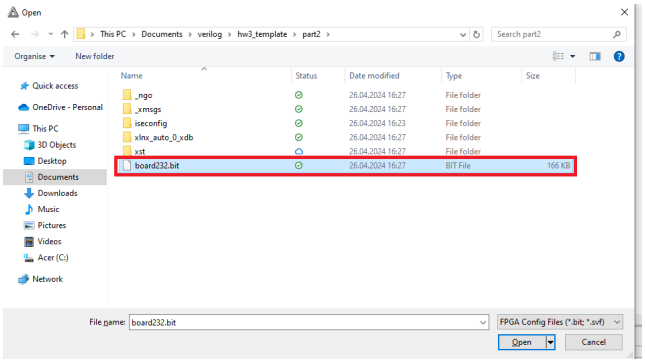

Figure 8: Selecting the "board232.bit" file.

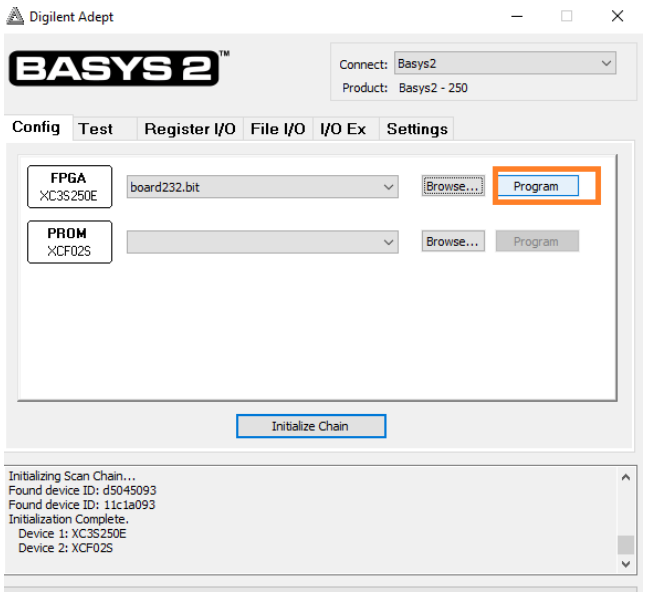

Figure 9: Programming the FPGA board.

5.1 Deliverables

- Implement your module in a single Verilog file. Upload only **lab3_2.v** file to ODTUClass system. Do **NOT** submit your testbenches, bit files or other project files. You can share your testbenches on ODTUClass discussion page.
- Submit the file through the ODTUClass system before the deadline given at the top.
- Use the ODTUClass discussion for any questions regarding the homework.
- This is individual work, any kind of cheating is not allowed.
- Implementations will be tested using custom test benches in a bloc-kbox fashion. We will use **Xilinx** and **iverilog** to conduct these tests. Please make sure that your code compiles over these tools.

- for iVerilog compilation please use **-g2005** flag, it is necessary in order to align the iVerilog tool with the Verilog standard that Xilinx uses.

6 Grading Criteria

Each part of this assignment is worth 50 points, totaling 100 points for the entire homework.

Part 1: For this part, your task involves simulating the required circuits using test benches. The grade for this part is based entirely on the correctness of your **simulations**. Each test case will examine specific outputs (i.e., Q_0, Q_1, and Z). To receive credit for a test case, **all outputs must exactly match the expected results** as specified in the provided tester files. This part will employ **black-box testing** through the test benches.

Part 2: In contrast to Part 1, the second part of the assignment requires your implementations to be **validated both through simulation test benches and physically on FPGA boards**. Each test case for this part will evaluate a subset of the six required outputs. The total points for each test case will be divided equally among these output tests, allowing for **partial credit for each correctly implemented output**. Similarly, FPGA board tests will follow the same methodology, with partial credits awarded where appropriate.

Important Note: The test benches provided are not exhaustive. Successfully passing all the provided test cases does not guarantee full credit, as they do not cover all possible scenarios. You are encouraged to extend these test benches.

7 Plagiarism

For any questions or discussions, if they do not contain specific code snippets etc., please use the discussion thread on `ODTUClass` so that everyone can benefit and be kept up-to-date. In such cases where you need to share specific code/pseudo code (anything not abstract regarding your solution), please do not hesitate to reach out via e-mail. You are encouraged to answer each other's questions and discuss among yourselves provided you do not share any code etc. Such actions of sharing are not to be tolerated. Similarly, submitting someone else's work in part or whole is a direct violation of academic integrity and will be subject to disciplinary action. This also holds for online/AI sources, including Google, ChatGPT, and Copilot. **Make use of them responsibly** - refrain from generating the code you are asked to implement. Remember that we also have access to these tools, making it easier to detect such cases. Your implementations will be subject to similarity checks through advanced tools and those that show high levels of similarity may be considered instances of plagiarism. **In short, please do not resort to practices violating your integrity - we are here to help.**