Assignment 3

Part 1: testbenches

Remarks:

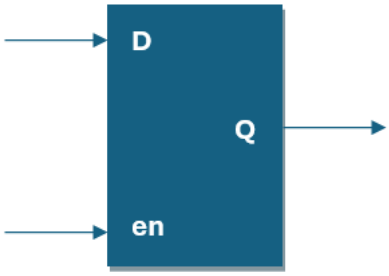
- **it is advised that the testbenches are a mix of randomization and directed testing with realistic inputs, to check for full functionality.**
- **Apply self-checking task for at least one of the designs below.**

Q1) Using the ALU (Arithmetic Logic Unit) designed in assignment 2, use Verilog to create a testbench for this design to test the functionality of the design.

The testbench should include at least 100 testcases.

Q2) Design and test the following D-latch.

Input	Output
D	Q
en	-



Truth Table		
En	D	Q
0	X	0
1	D	D

Make sure to create a testbench for this design with at least 100 test cases.

Part 2: Sequential

Q3) Design a Flip-Flop module that can function as either a D Flip-Flop or a T Flip-Flop based on a parameter FF_TYPE.

- **If FF_TYPE = DFF: The module should behave as a D Flip-Flop.**
- **If FF_TYPE = TFF: The module should behave as a T Flip-Flop.**
- **The Flip-Flop should have an active-low reset (rst_n).**
- **The module should provide both Q and Qbar (complement of Q) as outputs.**
- **Parameters:**
    1. **FF_TYPE: dafualt value = “DFF”**
- **Inputs:**
    1. **clk: Clock signal.**
    2. **rst_n: Active-low reset**
    3. **D: Data input**
- **Outputs:**
    1. **Q: Flip-Flop output.**
    2. **Qbar: Complement of Q.**

Q4) Draw the schematic for the following Verilog designs

```
module reg_blocking1 (
    input clk, rst, in,
    otput reg out
);
    reg q, q2;
    always @(posedge clk) begin
        if (rst) begin
            q2  = 0;
            q1  = 0;
            out = 0;
        end else begin
            q2  = q1;
            q1  = in;
            out = q2;
        end
    end
endmodule
```

```
module reg_non_blocking1 (
    input clk, rst, in,
    otput reg out
);
    reg q, q2;
    always @(posedge clk) begin
        if (rst) begin
            q2  <= 0;
            q1  <= 0;
            out <= 0;
        end else begin
            q2  <= q1;
            q1  <= in;
            out <= q2;
        end
    end
endmodule
```

```
module comb_blocking1 (
    input a, b, c, clk,
    output reg y
);
    always @(posedge clk) begin
        x = a & b;
        y = x | c;
    end
endmodule
```

```
module comb_non_blocking1 (
    input a, b, c, clk,
    output reg y
);
    always @(posedge clk) begin
        y <= x | c;
        x <= a & b;
    end
endmodule
```

Note: you can sketch the answer on a paper and add it to the assignment PDF

Submission Guidelines:

- **The submitted file must be a PDF file.**
- **Name the file in the following format: StudentName _assignment3.pdf**
- **The PDF must include:**
    - **Verilog design codes, testbench and the do file for part 1**
    - **Verilog design codes for part 2 (Q3)**
    - **Sketches of the last question**
    - **Waveform snippets from QuestaSim showing test results of each question**