Sequential Logic Design

- Design the following circuits using Verilog **and create a testbench** for each design to check its functionality. **Create a do file for question 1 &2. Run QuestaLint and make sure to have 0 errors. Generate schematics for questions 3, 4 and 6 using QuestLint. Lint demo here.**

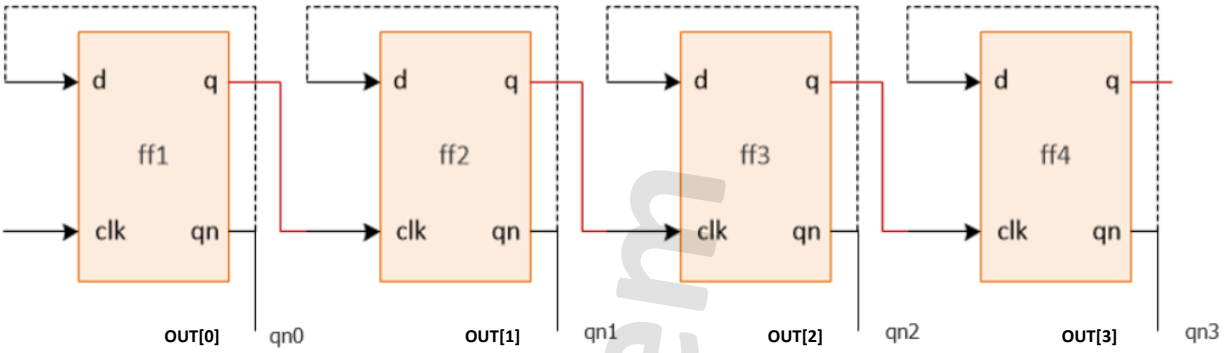
- Testbenches are advised to be a mix between randomization and directed **verification taken into consideration realistic operation** for the inputs. Please follow the testbench instructions for each question.

1) Write a self-checking testbench for question 2 in lab2 for the DFF with clock enable and PRE control signal.
2) Write a testbench for question 5 part C in assignment 2. Test the parameterized asynchronous FlipFlop using 2 testbenches, testbench 1 that overrides the design with FF_TYPE = "DFF" and the testbench 2 overrides parameter with FF_TYPE = "TFF".
    - Testbench 1 should instantiate the design of question 5 part B assignment 2 as a golden model to check for the output of the parameterized design with FF_TYPE = "DFF"
        - Use a do file to run the simulation
    - Testbench 2 should instantiate the design of question 5 part A assignment 2 as a golden model to check for the output of the parameterized design with FF_TYPE = "TFF"
        - Use a do file to run the simulation
3) Implement BCD up counter (MOD 10 counter), where the counter has 10 states. The counter will divide the clock frequency by 10. The following is the counter interface:
    - Inputs:
        i. Clk
        ii. Rst (async active high)
    - Outputs
        i. Clk_div10_out

Hint: Toggle the output based on the internal counter value so that the output frequency equals the input clock frequency divided by 10.

- Testbench should apply reset and remove it then add a delay of 100 clock cycles. Make sure output is correct from the waveform.

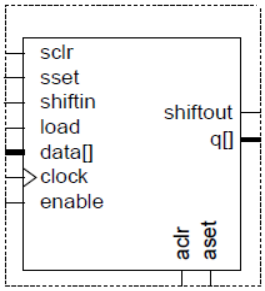
4) Implement the 4-bit Ripple counter shown below using structural modelling (Instantiate the Dff from question 5 part B assignment 2 where the output is taken from the qn as shown below). Implement a randomized testbench and check the output correctness from the waveform.
    - Inputs: clk, rstn
    - Outputs: [3:0] out



5) Implement the following Parameterized Shift register

1. Parameters

Name	Value	Description
LOAD_AVALUE	Integer > 0	Value loaded with aset is high
SHIFT_DIRECTION	"LEFT" or "RIGHT"	Direction of the shift register. Default = "LEFT"
LOAD_SVALUE	Integer > 0	Value loaded with sset is high with the rising clock edge
SHIFT_WIDTH	Integer > 0	Width of data[] and q[] ports



Default value for LOAD_AVALUE and LOAD_SVALUE is 1. SHIFT_WIDTH default value is 8.

2. Ports

Name	Type	Description
sclr	Input	Synchronous clear input. If both sclr and sset are asserted, sclr is dominant.
sset		Synchronous set input that sets q[] output with the value specified by LOAD_SVALUE. If both sclr and sset are asserted, sclr is dominant.
shiftin		Serial shift data input
load		Synchronous parallel load. High: Load operation with data[], Low: Shift operation
data[]		Data input to the shift register. This port is SHIFT_WIDTH wide
clock		Clock Input
enable		Clock enable input
aclr		Asynchronous clear input. If both aclr and aset are asserted, aclr is dominant.
aset		Asynchronous set input that sets q[] output with the value specified by LOAD_AVALUE. If both aclr and aset are asserted, aclr is dominant.
shiftout	Output	Serial Shift data output
q[]		Data output from the shift register. This port is SHIFT_WIDTH wide

Notes:

1- Enable signal enables the synchronous operation of the design. Enable signal is dominant over the synchronous control signals "sclr and sset". The synchronous control signals "sclr and sset" are dominant over the load signal.

2- shiftout output represents the bit removed of the register and not the most significant bit.

Testbench for this design will be as follows:

1. Parameter Overrides

- LOAD_AVALUE = 2
- LOAD_SVALUE = 4
- SHIFT_DIRECTION = "LEFT"
- SHIFT_WIDTH = 8

2. Stimulus Generation (Initial Block)

2.1 Verify Asynchronous Clear (aclr) Functionality

- Drive aclr and aset to **1**.
- Randomize the remaining inputs inside a for loop.
- Wait for the **negative edge of the clock**.
- Add a condition for **output q** to enable self-checking.

2.2 Verify Asynchronous Set (aset) Functionality

- Drive aclr to **0** and aset to **1**.
- Randomize the remaining inputs inside a for loop.
- Wait for the **negative edge of the clock**.
- Check **output q** for correctness.

2.3 Verify Synchronous Clear (sclr) Functionality

- Drive aset and aclr to **0**, sclr and sset to **1**.
- Randomize the remaining inputs inside a for loop.
- Wait for the **negative edge of the clock**.
- Add a condition for **output q** to enable self-checking.

2.4 Verify Synchronous Set (sset) Functionality

- Drive aset and aclr to **0**, sclr to 0, and sset to **1**.
- Randomize the remaining inputs inside a for loop.
- Wait for the **negative edge of the clock**.
- Add a condition for **output q** to enable self-checking.

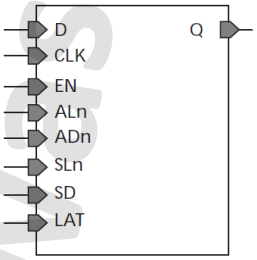
2.5 Verify Load Functionality

- Drive load to **1** and deactivate **sync and async control signals**.
- Randomize the remaining inputs inside a for loop.
- Wait for the **negative edge of the clock** then add condition for **output q** to enable self-checking.

2.6 Verify Shifting Functionality

- Deactivate **control signals and load signal**
- Drive the remaining inputs in a **realistic manner** & Check the waveform for **shifting functionality**

6) Implement the following SLE (sequential logic element). This design will act as flipflop or latch based on the LAT signal as demonstrated in the truth table.



Input		Output
Name	**Function**	
D	Data	
CLK	Clock	
EN	Enable	
ALn	Asynchronous Load (Active Low)	Q
ADn*	Asynchronous Data (Active Low)	
SLn	Synchronous Load (Active Low)	
SD*	Synchronous Data	
LAT*	Latch Enable	

Truth Table

ALn	ADn	LAT	CLK	EN	SLn	SD	D	Q_{n+1}
0	ADn	X	X	X	X	X	X	!ADn
1	X	0	Not rising	X	X	X	X	Qn
1	X	0	↑	0	X	X	X	Qn
1	X	0	↑	1	0	SD	X	SD
1	X	0	↑	1	1	X	D	D
1	X	1	0	X	X	X	X	Qn
1	X	1	1	0	X	X	X	Qn
1	X	1	1	1	0	SD	X	SD
1	X	1	1	1	1	X	D	D

Testbench should start with activating ALn then deactivating it. Then for simplicity we keep the LAT to 0 and in a repeat block randomize all inputs. Then drive LAT to 1 and in a repeat block randomize all inputs. Check the functionality correctness of each input from the waveform.

Deliverables:

The assignment should be submitted as a PDF file with this format <your_name>_Assignment3 for example Kareem_Waseem_Assignment3

Note that your document should be organized as 6 sections corresponding to each design above, and in each section, I am expecting the Verilog code, and the waveforms snippets and questalint snippet of 0 errors and schematics for questions 3, 4 and 6.