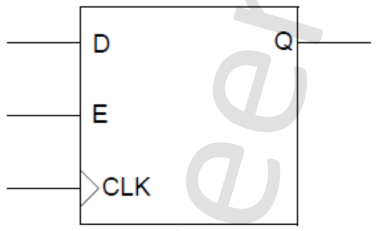
Sequential Logic Design

- Design the following circuits using Verilog **and create a testbench** for each design to check its functionality. **<u>Create a do file for question 5 only.</u>**

- Testbenches are advised to be a mix between randomization and directed testing taken into consideration realistic operation for the inputs. Apply self-checking condition in the testbench for at least one of the design below.

1) Implement D-Type Flip-Flop with active high Enable.

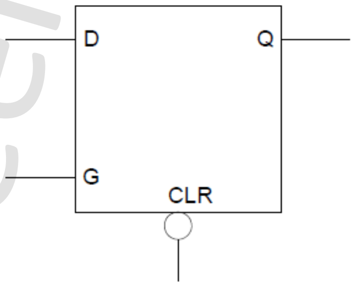


Input	Output
D, E, CLK	Q

Truth Table

E	CLK	D	Q_{n+1}
0	X	X	Q_n
1	not Rising	X	Q_n
1	↑	D	D

2) Implement Data Latch with active low Clear



Input	Output
CLR, D, G	Q

Truth Table

CLR	G	D	Q
0	X	X	0
1	0	X	Q
1	1	D	D

3) Implement the following latch as specified below

Parameters

LAT_WIDTH: Determine the width of input data and output q

Ports

Name	Type	Description
aset	Input	Asynchronous set input. Sets q[] output to 1.
data[]		Data Input to the D-type latch with width LAT_WIDTH
gate		Latch enable input
aclr		Asynchronous clear input. Sets q[] output to 0.
q[]	Output	Data output from the latch with with LAT_WIDTH

If both aset and aclr are both asserted, aclr is dominant.

4)

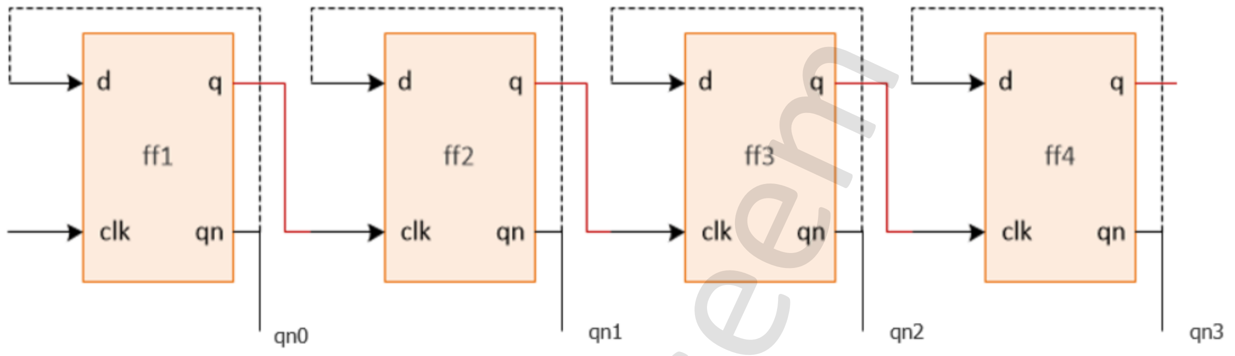
A. Implement T-type (toggle) Flipflop with active low asynchronous reset. T-Flipflop has input t, when t input is high the outputs toggle else the output values do not change.
   - Inputs: t, rstn, clk
   - Outputs: q, qbar

B. Implement Asynchronous D Flip-Flop with Active low reset
   - Inputs: d, rstn, clk
   - Outputs: q, qbar

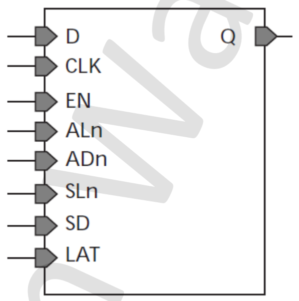
C. Implement a parameterized asynchronous FlipFlop with Active low reset with the following specifications.
   - Inputs: d, rstn , clk
   - Outputs: q, qbar
   - Parameter: FF_TYPE that can take two valid values, DFF or TFF. Default value = “DFF” Design should act as DFF if FF_TYPE = “DFF” and act as TFF if FF_TYPE = “TFF”. When FF_TYPE equals “DFF”, d input acts as the data input “d”, and when FF_TYPE equals “TFF”, d input acts the toggle input “t”.

D. Test the above parameterized Design 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 part B. as a golden model to check for the output of the parameterized design with FF_TYPE = “DFF”
   - Testbench 2 should instantiate the design of part A. as a golden model to check for the output of the parameterized design with FF_TYPE = “TFF”

5) Implement the 4-bit Ripple counter shown below using structural modelling (Instantiate the Dff from question 4 part B where the output is taken from the qn as shown below)
   - Inputs: clk, rstn;
   - Outputs: [3:0] out;



6) Implement the following SLE (sequential logic element)



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

***Note**: ADn, SD and LAT are static signals defined at design time and need to be tied to 0 or 1.

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

<u>Deliverables:</u>

1) The assignment should be submitted as a PDF file with this format <your_name>_Assignment3 for example Kareem_Waseem_Assignment3
2) Snippets from the waveforms captured from QuestaSim for each design with inputs assigned values and output values visible.

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