Counters & Shift Registers

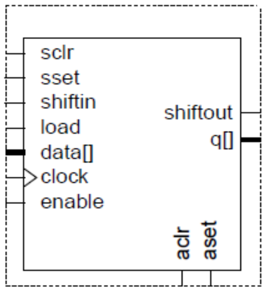
Design the following circuits using Verilog **and create a testbench** for each design to check its functionality. **Create a do file for question 2 only.**

- Testbenches are advised to be a mix between randomization and directed testing taken into consideration realistic operation for the inputs.

1) Implement the following Parameterized Shift register

- 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



- 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

2)

1. Implement 4-bit Ripple counter with asynchronous active low set using behavioral modelling that increment from 0 to 15
   - Input:
     - clk
     - set (sets all bits to 1)
     - out (4-bits)

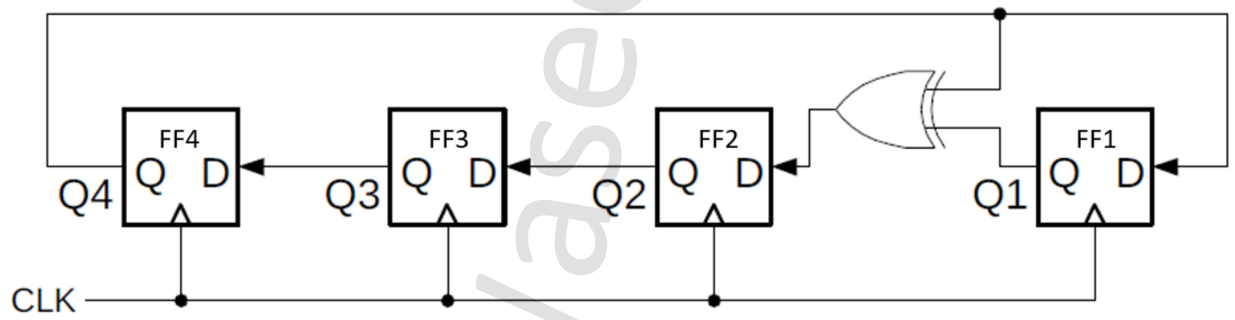
2. Use the structural counter done in the previous assignment as a golden reference.
3. Test the above behavioral design using a self-checking testbench
    — Testbench should instantiate the previous two designs

3) Extend on the previous counter done in question 3 to have extra 2 single bit outputs (div_2 and div_4). Hint: Observe the output bits of the "out" bus to generate the following

- div_2: output signal that acts as output clock with a frequency half the input clock signal.
- div_4: output signal that acts as output clock with a frequency quarter the input clock signal.

4) Implement the following Linear feedback shift register (LFSR)

- LFSR Inputs: clk, rst, set
- LFSR output: out (4 bits) where out[3] is connected to Q4, out[2] is connected to Q3, etc.



- LFSR can be used as a random number generator.
- The sequence is a random sequence where numbers appear in a random sequence and repeats as shown on the figure on the right

0001
0010
0100
1000
0011
0110
1100
1011
0101
1010
0111
1110
1111
1101
1001
0001

FF2, FF3, FF4 have the following specifications:

- D input
- Clk input
- Input async rst (active high) – resets output to 0
- Output Q

FF1 have the following specifications:

- D input
- Clk input
- Input async set (active high) – set output to 1
- Output Q

Note: the rst and set signals should be activated at the same time to guarantee correct operation

5) Implement N-bit parameterized Full/Half adder

- Parameters
  - WIDTH: Determine the width of input a,b, sum
  - PIPELINE_ENABLE: if this parameter is high then the output of the sum and carry will be available in the positive clock edge (sequential) otherwise the circuit is pure combinational, default is high
  - USE_FULL_ADDER: if this parameter is high then cin signal will be used during the cout and sum calculation from the input signals, otherwise if this parameter is low ignore the cin input, default is high
- Ports

Name	Type	Description
a	Input	Data input a of width determined by WIDTH parameter
b		Data input b of width determined by WIDTH parameter
clk		Clk input
cin		Carry in bit
rst		Active high synchronous reset
sum	Output	sum of a and b input of width determined by WIDTH parameter
cout		Carry out bit

<u>Deliverables:</u>

1) The assignment should be submitted as a PDF file with this format <your_name>_Assignment4 for example Kareem_Waseem_Assignment4
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 5 sections corresponding to each design above, and in each section, I am expecting the Verilog code, and the waveforms snippets