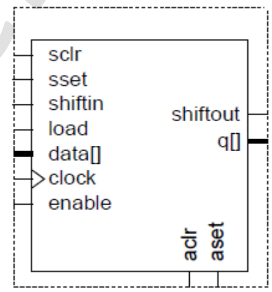
# Counters, Shift Registers & Memories

Design the following circuits using Verilog **and create a testbench** for each design to check its functionality

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 |

- Note that the synchronous control signals "sclr and sset" have dominance over the enable signal but the enable signal is dominant over the load signal.

Kareem Wasem

2)

1. Implement Asynchronous D Flip-Flop with Active low reset

   Inputs: d, rstn, clk

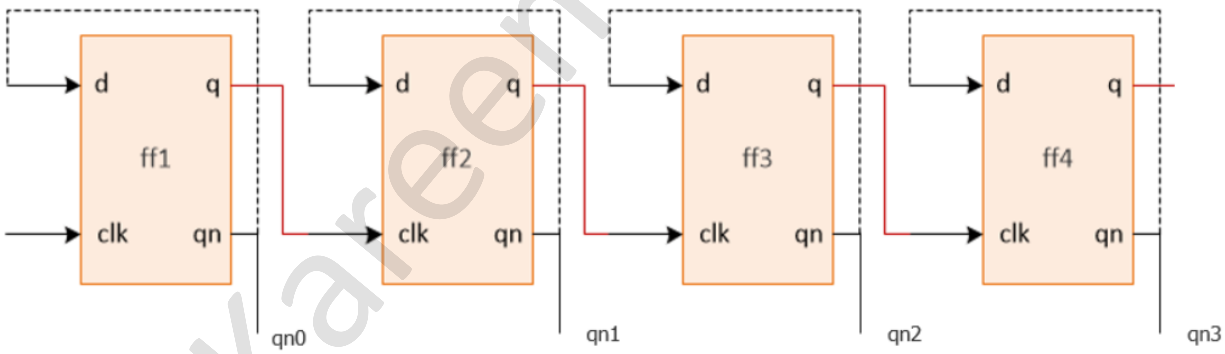
   Outputs: q, qbar

2. Implement 4-bit Ripple counter with asynchronous active low set using behavioral modelling that increment from 0 to 15
3. Implement the below 4-bit Ripple counter using structural modelling (Instantiate DFF previously designed in step 1). Ripple counter output “out” is connected to qn0, qn1, qn2, qn3 which are connected to the qn output of each DFF as shown below)

   Inputs: clk, rstn

   Outputs: [3:0] out

4. Test the above structural design using a testbench
   — Testbench should instantiate the previous two counters designed in steps 2 and 3
   — Consider the behavioral design as the golden model and check the functionality of the structural design



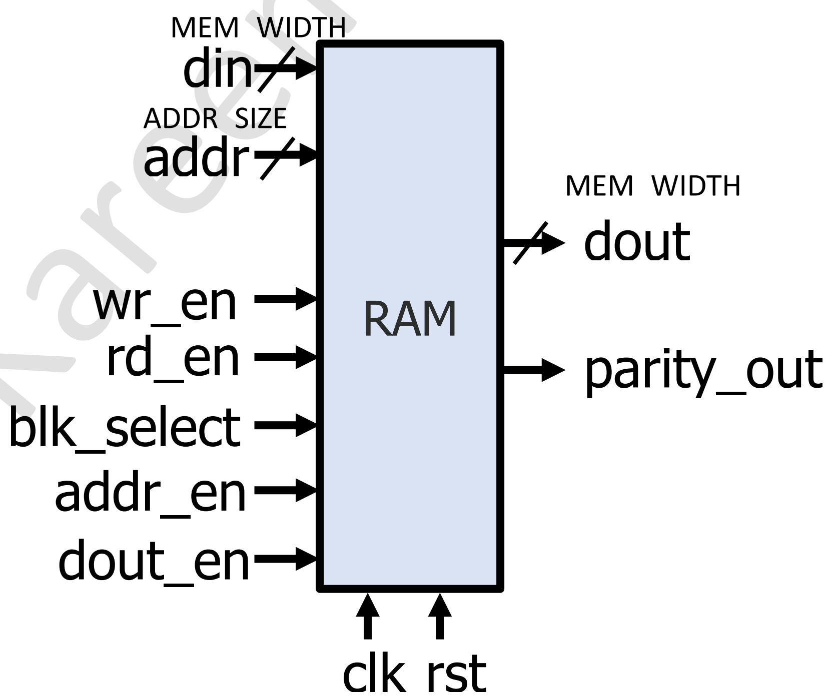

Kareem Waseem

3) Implement the following single port synchronous write/read

- Parameters

| Name | Description | Default values |
|---|---|---|
| MEM_WIDTH | Data in/out and memory word width | 16 |
| MEM_DEPTH | Memory depth | 1024 |
| ADDR_SIZE | Address size based upon the memory depth | 10 |
| ADDR_PIPELINE | If "TRUE" then the address should be pipelined before writing/reading the RAM, if "FALSE" then the address input will be assigned directly to the RAM's address port | FALSE |
| DOUT_PIPELINE | If "TRUE" then the data out should be pipelined, if "FALSE" then the output will be out of the RAM directly | TRUE |
| PARITY_ENABLE | If the parameter value is 1 then the parity should be calculated and assigned to parity_out port, if the parameter is 0 then the parity_out port should be tied to 0 | 1 |

- Added ports functionality
  - addr_en: enable signal for the flipflop that pipelines the address
  - dout_en: enable signal for the flipflop that pipelines the data out
  - parity_out: calculates the parity on the dout bus




Kareem Waseem

**Extra Problems to be solved:**

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 pipelined 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 |
| parity_out | | Parity bit that should be 1 if the if the number of 1's in the sum and cout is odd (instantiate the parity module inside the full adder) |

Kareem Waseem