**ECE-111 Advanced Digital Design Projects**

**Homework-4:**

* Develop SystemVerilog RTL model for an n-bit Barrel Shifter using 2x1 Mux behavioral model
* Same Barrel Shifter RTL model should support left shift, right shift, left rotate and right rotate operation
* Use 2x1 Mux behavioral RTL model (will be provided) to design barrel shifter (**Do not use nx1 Mux model**)
* Synthesize 4-bit Barrel Shifter and review synthesis results (resource usage and RTL netlist/schematic)
* Run simulation using testbench provided and review waveform to confirm :
* left shift, right shift, left rotate and right rotate operation of barrel shifter RTL model behavior
* Assume below mentioned primary port names and SystemVerilog RTL module name as **barrel shifter**

Barrel Shifter

**shift\_value[1:0]**

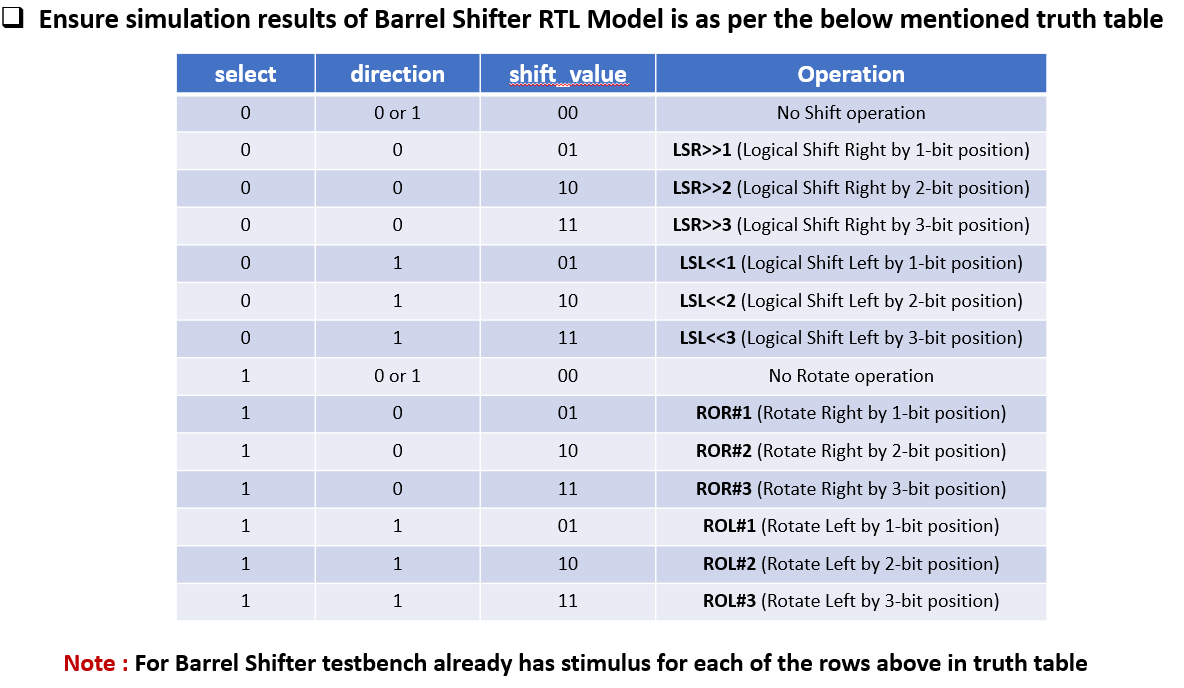
**select**

**dout[n-1:0]**

**din[n-1:0]**

**direction**

* **Primary Ports for Barrel Shifter**
* **select : to select between shift or rotate operation**
* select == 0 for shift operation
* select == 1 for rotate operation
* **din : n-bit input data**
* **dout : n-bit output data**
* **direction : move bits in either left or right**
* direction == 0, move bit to right
* direction == 1, move bit to left
* **shift\_value : bit positions to be shifted**
* shift\_value == 00, no shift operations
* shift\_value == 01, move bits by 1-bit position
* shift\_value == 10, move bits by 2-bit positions
* shift\_value == 11, move bits by 3-bit positions

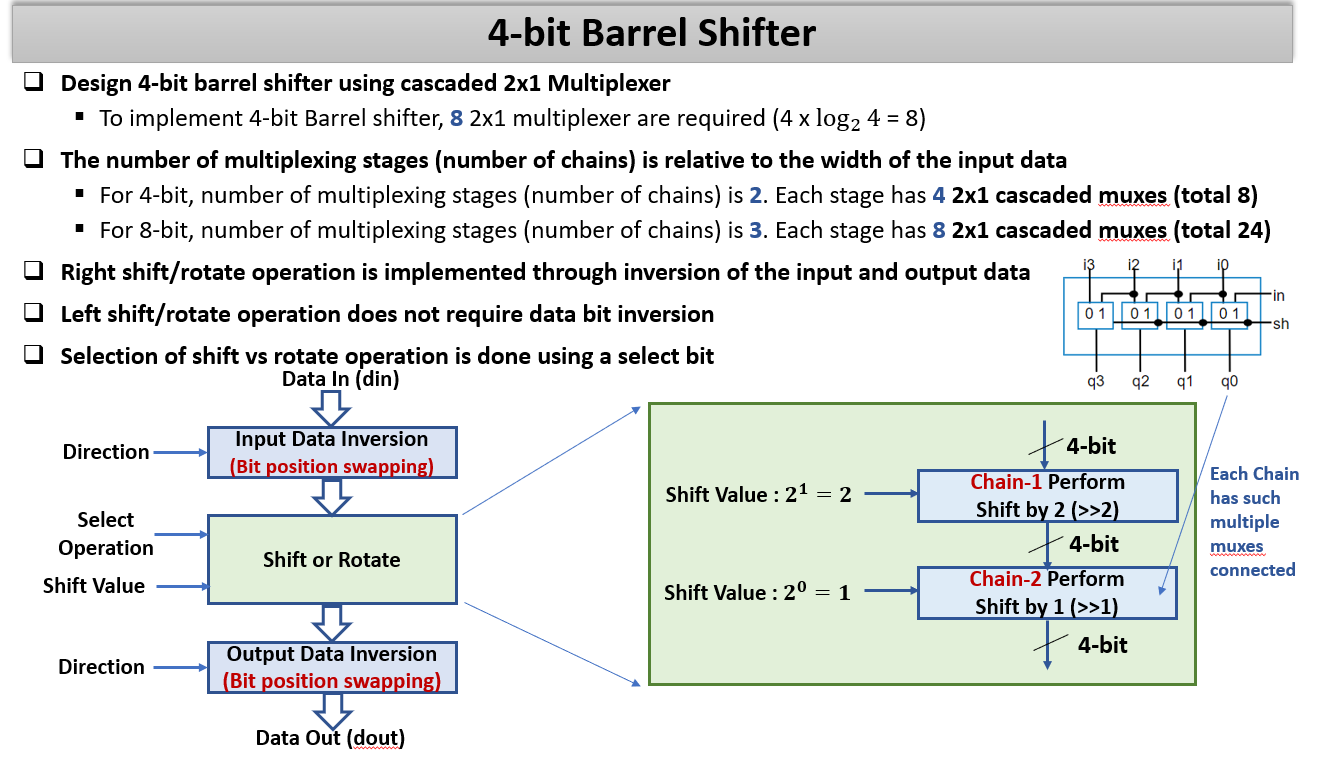


**Homework Submission :**

**Verilog code for design and test bench (attach .sv files)**

**Submit report (PDF file) which should include:**

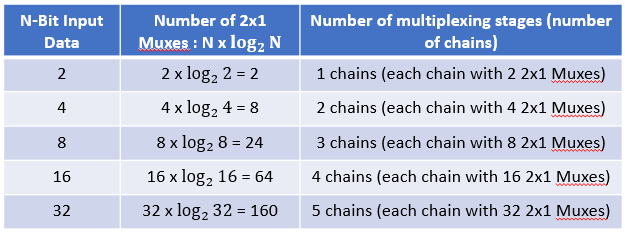
* Synthesis resource usage and schematic generated from RTL netlist viewer
* Simulation snapshot and explain simulation result to confirm it works as a barrel shifter with each type of operations (one example of each left shift, right shift, rotate left and rotate right)
* Resource usage explanation and post mapping schematic is optional to submit.

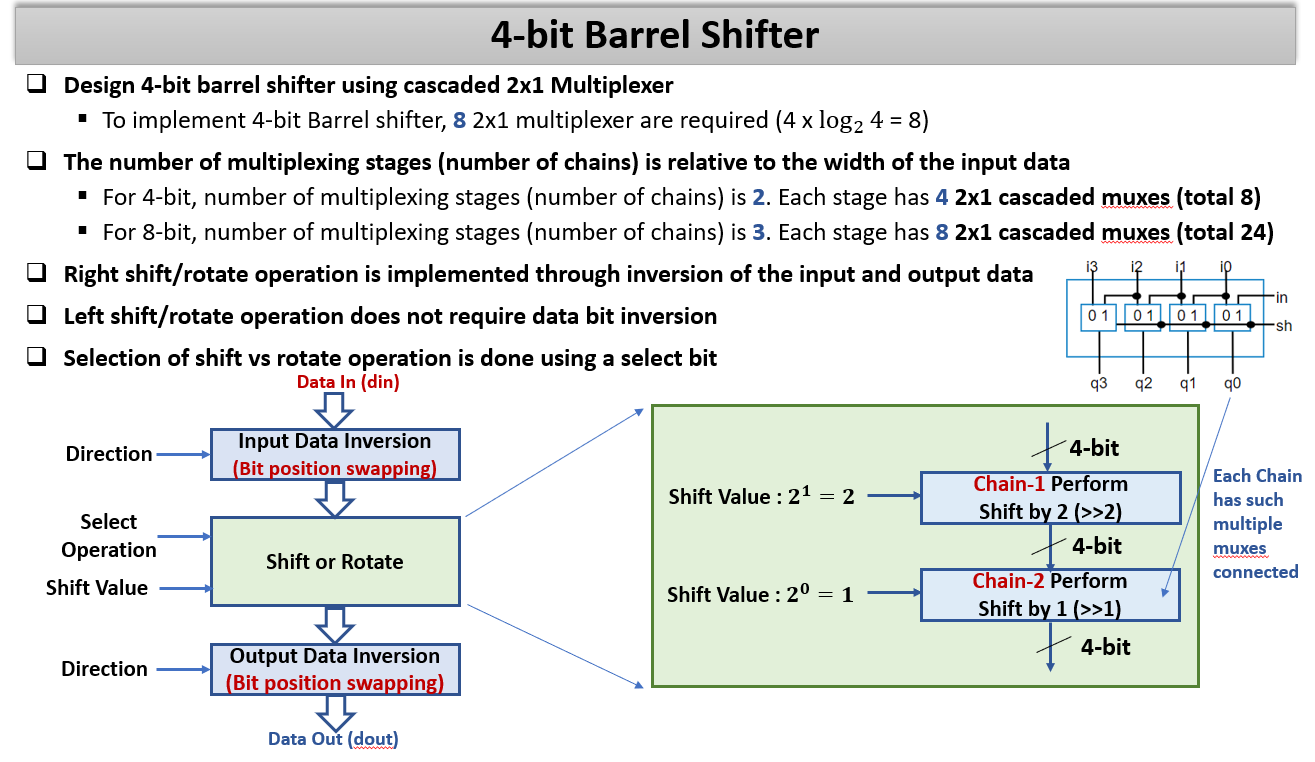


**Note : CORRECTION: right shift/rotate does NOT require inversion; left does**

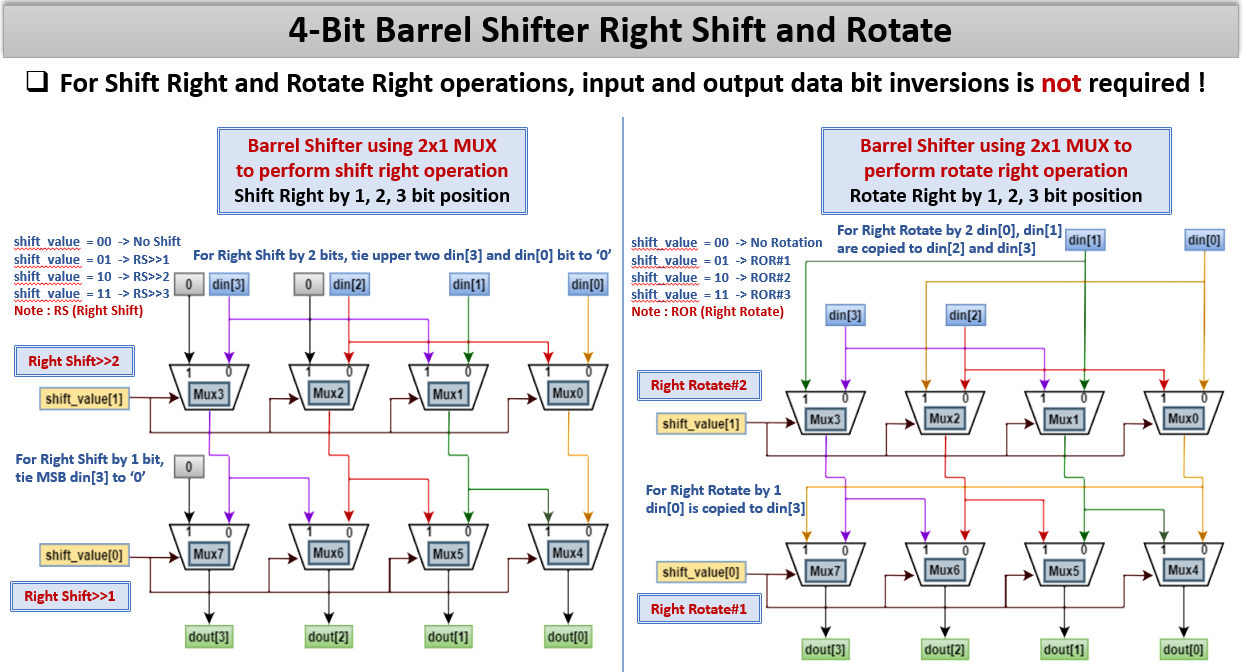
**Remember bit inversion here does not mean negation of input values, it means swapping bit positions !**

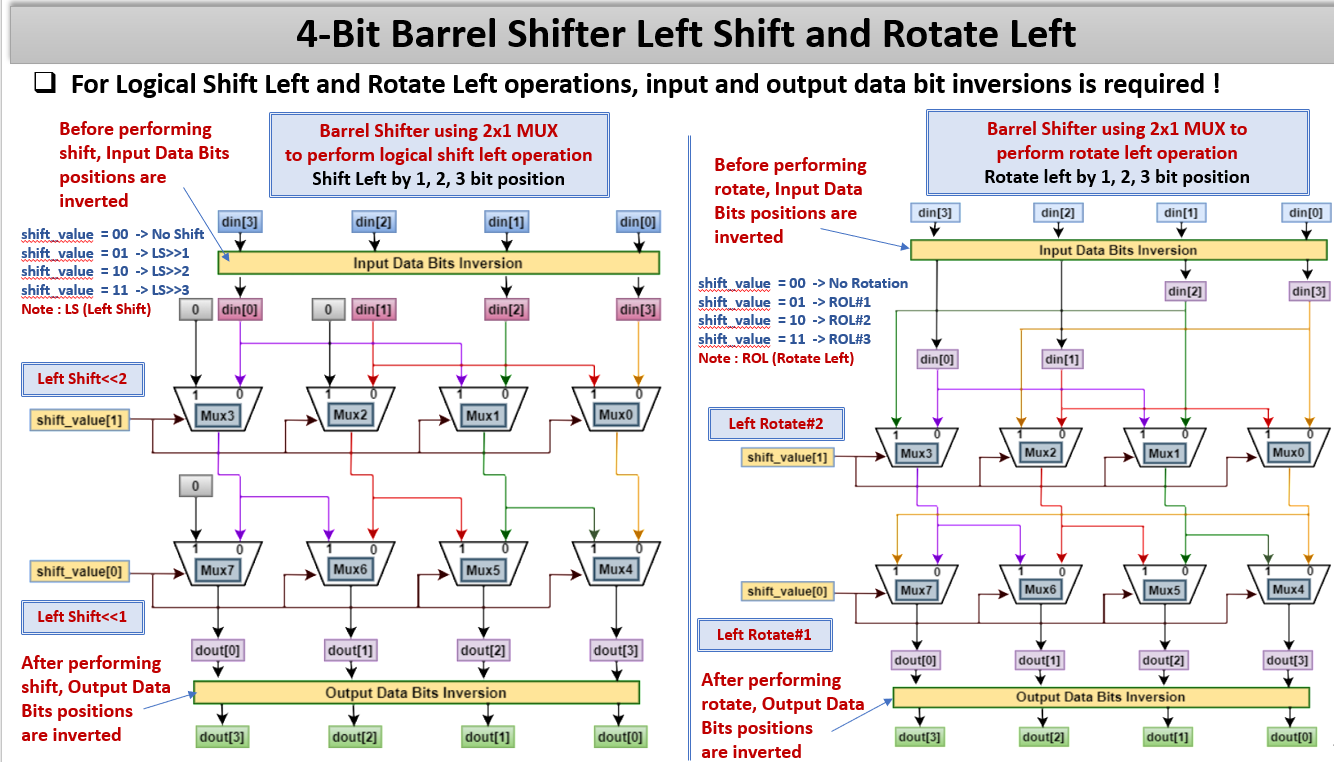
**Example : if din = 0011 then after bit data inversion din should be = 1100**





**Note : CORRECTION: right shift/rotate does NOT require inversion; left does**





***NOTE***: The hints says “0: left, 1: right” for direction. Ignore this. Use 0 for right shift, as in the instructions.

