I. Pre-Lab3_1 (4-bit synchronous binary up counter)

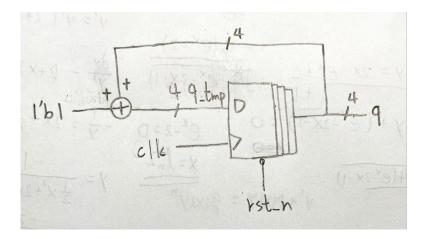
Design Specification

IO: Input: clk, rst_n. Output: [3:0] q.

Design Implementation

First, according to binary up counter, output add 1 after passing D-flip flop. So I declare a variable [3:0] q_tmp to store the result temporarily after adding 1. Then I can construct the logic diagram:

Logic diagram:



Finally, construct Verilog RTL code for the binary up counter:

```
22
23
    module counter(
24
        output reg [3:0] q,
25
        input clk,
        input rst_n
26
27
        );
        reg [3:0] q_tmp;
28
29 Ė
        always@*
30
             q_tmp = q + 1'b1;
        always@(posedge clk or negedge rst_n)
31
             if(~rst_n)
32
                 q <= 4'd0;
33
34
             else
35
                 q <= q_tmp;
36 🍐 endmodule
37
```

Simulation:

| | | 0.004 ns | | | | | | | | | | | | | | | | |
|------------|-------|----------|---|-------|---|-------|-----|-------|-----|-------|----------|----------|---|--------|---|--------|-----|--------|
| Name | Value | 0 ns 1 | | 20 ns | | 40 ns | | 60 ns | | 80 ns | | 100 ns | | 120 ns | | 140 ns | | 160 ns |
| l‰ clk | 1 | | | | | | | | | | | | | | | | | |
| 🎼 rst_n | 0 | | | | | | | | | | | | | | | | | |
| 🖪 📲 q[3:0] | 0 | | 1 | 2 | 3 | 4 | X s | 6 | X 1 | 8 | <u> </u> | <u>a</u> | Ъ | ¢. | A | e | ∕_f | X O |
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |

II. Pre-Lab3_2 (ringer counter)

Design Specification

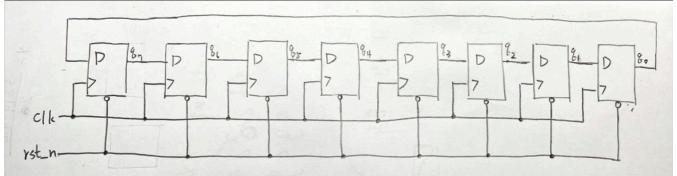
IO: Input: clk, rst_n. Output: [7:0] q.

Design Implementation

Because this is a ring counter, I connect each flip flop by following method

 $\begin{array}{l} q[6] <= q[7];\\ q[5] <= q[6];\\ q[4] <= q[5];\\ q[3] <= q[4];\\ q[2] <= q[3];\\ q[1] <= q[2];\\ q[0] <= q[1];\\ q[7] <= q[0];\\ \end{array}$ and if rst_n = 1, D-flip flop should be 10010110.

Logic diagram:



Finally, construct Verilog RTL code for the ringer counter:

```
module shifter(
   output reg [7:0] q,
   input clk,
   input rst_n
   );
   always@(posedge clk or negedge rst_n)
   if(~rst_n)
       q <= 8'b10010110;
   else
   begin
       q[б] <= q[7];
       q[5] <= q[6];
       q[4] <= q[5];
       q[3] \iff q[4];
       q[2] \ll q[3];
       q[1] \ll q[2];
       q[0] \ll q[1];
        q[7] \ll q[0];
   end
        I
endmodule
```

Simulation:

| Sintanation. | | | | | | | | | | | | | | | | _ | |
|--------------|----------|----------|------|-------|--------|----------|------|-------|------|---------|------|--------|------|----------|------|--------|-----|
| | | 0.002 ns | | | | | | | | | | | | | | | |
| Name | Value | 0 ns | | 20 ns | | 40 n.s. | | 60 ns | | 80 n.s. | | 100 ns | | 120 ns | | 140 ns | |
| 16 clk | 1 | | | | | | | ~ | | ~ | | | | | | | H |
| 🐚 rst_n | o | | | | | <u> </u> | | | | | | | | <u> </u> | | | - |
| 🔳 📲 q[7:0] | 10010110 | 1001 | 0010 | 0101 | X 1011 | 0110 | 1101 | 1010 | 0100 | 1001 | 0010 | 0101 | 1011 | 0110 | 1101 | 1010 | 010 |
| Vie [7] | 1 | | | | | | | | 1 | | 1 | | | | | | |
| Ալ6] | 0 | | | | 1 | | | | | | | | | | | | |
| Ալ [5] | 0 | | | | | | 1 | | 1 | | | | | | | | |
| V# [4] | 1 | | | | | | | | | | 1 | | | | | | |
| 14 [3] | 0 | | | | | | 1 | | | | | | | | | | |
| Via [2] | 1 | | | | | | | | 1 | | | | | | | | |
| Via [1] | 1 | | | | 1 | | | | | | 1 | | | | | | |
| ۳. [0] | 0 | | | | | | 1 | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |