Pre-Lab3 Report

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

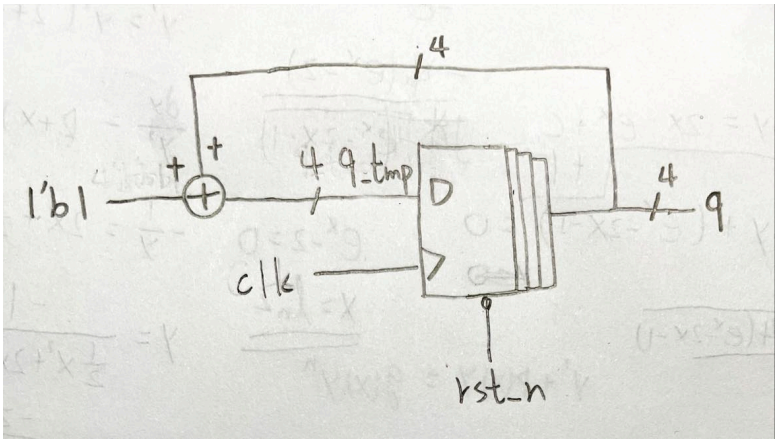
Design Specification

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:

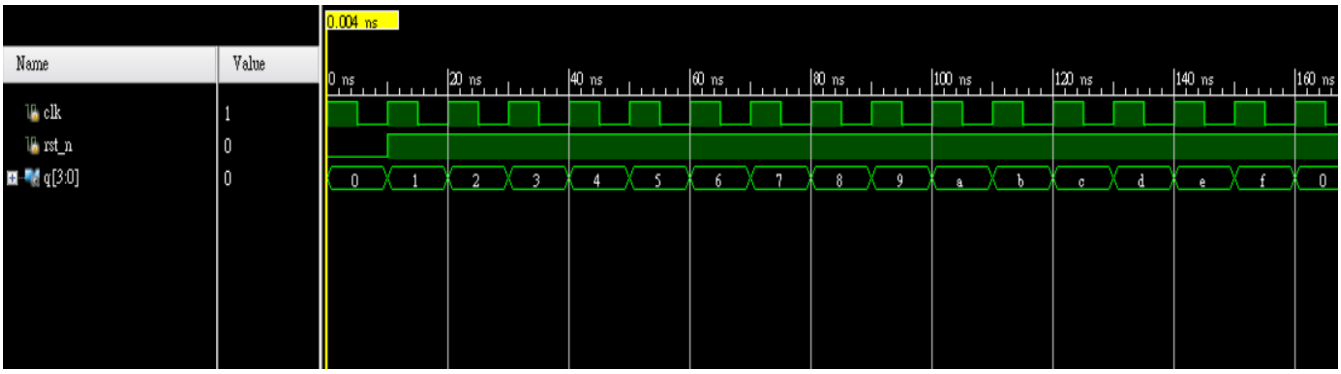
Logic diagram:



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

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

Simulation:



Ⅱ. Pre-Lab3_2 (ringer counter)

Design Specification

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

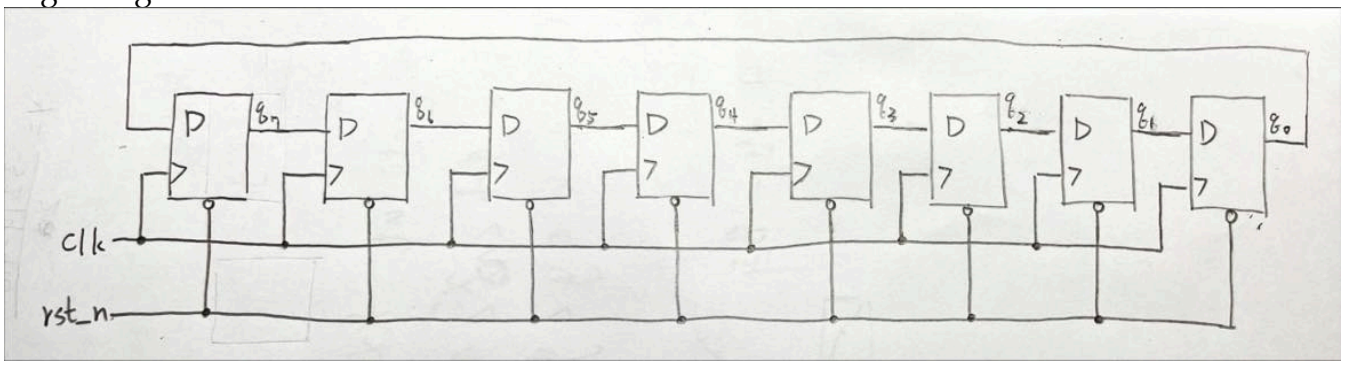
Design Implementation

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

```
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];
```

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[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

endmodule
```

Simulation: