ser2par.v

```
`timescale 1ns / 1ns

module ser2par (
    input clk,
    input rst_n,
    input din,
    output [7:0] dout,
    output vldout);

    reg [7:0] data_out;
    assign dout = data_out;
    reg valid_data;
    assign vldout = valid_data;
    integer count = 0;

    always @(posedge clk) begin
        if (!rst_n) begin
          data_out <= 0;
          valid_data <= 1'b0;
          count = 0;   #7
        end
        else begin
            count +=1;   #7
            if (count == 8) begin
                count = 0;
                data_out <= {data_out[6:0], din};
                valid_data <= 1'b1;
            end
            else if(valid_data == 1) begin #4
                valid_data <= 0;
                data_out <= {7'b0, din};
            end
            else begin
               data_out <= {data_out[6:0], din};
            end
        end
    end
endmodule
```

mynotes.v

```
`timescale 1ns / 1ps

module ser2par (
    input clk,
    input rst_n,
    input din,
    output reg [7:0] dout, #2
    output reg vldout      #2
);
    reg [2:0] count; #6
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            dout <= 0;
            vldout <= 1'b0;
            count <= 0;
        end else begin
            dout <= {dout[6:0], din};
            count <= count + 1;
            vldout <= (count == 7);
        end
    end
endmodule
```

#3. most places use async reset in always **(negedge rst_n)**
#4. if(aaa == 1'b1) is equal to if (aaa)
#5. integer count = 0;   <<< works only in FPGA.
#6. reg[2:0] automatically wraps to zero.
#7. in flops, must use "<=", "=' not synthesizable
#8. my personal pref: **end else if () begin** in the same line

#1. No need to use FPGA style --> where output cannot be not internal signal.
#2. output is fine to be reg