

```
// -----
```

```
module shift_register #(
    parameter N = 10,
    parameter M = 8
) (
    input wire [N-1:0] din,
    input wire        clk,
    input wire        rst,
    input wire        enable,
    input wire        load,
    input wire [N-1:0] value,
    output wire [N-1:0] dout
);

    reg [N-1:0] sr [M-1:0];
    integer i;

    wire [N-1:0] sr_0;

    assign sr_0 = sr[0];

    always @(posedge clk, posedge rst) begin

        if (rst) begin
            for (i=0;i<M;i=i+1) begin
                sr[i] <= {N{1'b0}};
            end
        end
        else begin
            if (enable) begin
                if (load) begin
                    for (i=0;i<M;i=i+1) begin
                        sr[i] <= value;
                    end
                end
                else begin
                    sr[0] <= din;
                    for (i=1; i<M;i=i+1) begin
                        sr[i] <= sr[i-1];
                    end
                end
            end
        end

        assign dout = sr[M-1];
    endmodule
```

```
// -----
```

```
module seq_det (
    input wire clk,
    input wire rst,
    input wire din,
    output wire det
);

    localparam S3 = 2'b00;
    localparam S1 = 2'b01;
    localparam S2 = 2'b10;
    localparam S0 = 2'b11;

    reg [1:0] state;
```

```
// -----  
// One hot  
// -----  
// localparam S3 = 4'b0001;  
// localparam S1 = 4'b0010;  
// localparam S2 = 4'b0100;  
// localparam S0 = 4'b1000;  
  
// reg [3:0] state;  
// -----  
  
always @(posedge clk, posedge rst) begin  
    if (rst) begin  
        state <= S0;  
    end  
    else begin  
        case(state)  
            S0: if(din) begin  
                state <= S1;  
            end  
            S1: if (din) begin  
                state <= S2;  
            end  
            else begin  
                state <= S0;  
            end  
            S2: if (!din) begin  
                state <= S3;  
            end  
            S3: if (din) begin  
                state <= S1;  
            end  
            else begin  
                state <= S0;  
            end  
        endcase  
    end  
  
    assign det = (state==S3);  
  
    // -----  
    // Otra opción  
    // -----  
    // reg [1:0] next_state;  
    //  
    // always @(posedge clk, posedge rst) begin  
    //     if (rst) begin  
    //         state <= S0;  
    //     end  
    //     else begin  
    //         state <= next_state;  
    //     end  
    // end  
    //  
    // always @(*) begin  
    //     case(state)  
    //         S0: if(din) bgin  
    //             next_state = S1;  
    //         end  
    //         else begin  
    //             next_state = S0;  
    //         end  
    //         S1: if (din) begin  
    //             next_state = S2;  
    //         end  
    //         else begin  
    //             next_state = S0;  
    //         end  
    //     end  
    // end
```

```
//      S2: if (!din) begin
//          next_state = S3;
//      end
//      else begin
//          next_state = S2;
//      end
//      S3: if (din) begin
//          next_state = S1;
//      end
//      else begin
//          next_state = S0;
//      end
//  endcase
// end
// -----

// -----
// Otra opción
// -----
// always @(*) begin
//     next_state = state;
//     case(state)
//         S0: if(din) begin
//             next_state = S1;
//         end
//         S1: if (din) begin
//             next_state = S2;
//         end
//         S2: if (!din) begin
//             next_state = S3;
//         end
//         S3: if (din) begin
//             next_state = S1;
//         end
//     endcase
// end
// -----
```

endmodule