Fsm hdlc

Synchronous HDLC framing involves decoding a continuous bit stream of data to look for bit patterns that indicate the beginning and end of frames (packets). Seeing exactly 6 consecutive 1s (i.e., `01111110`) is a "flag" that indicate frame boundaries. To avoid the data stream from accidentally containing "flags", the sender inserts a zero after every 5 consecutive 1s which the receiver must detect and discard. We also need to signal an error if there are 7 or more consecutive 1s.

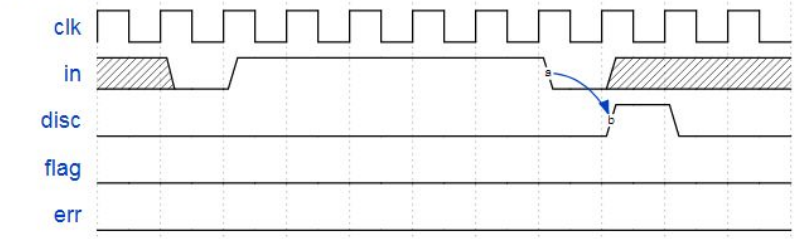
Create a finite state machine to recognize these three sequences:

- `0111110`: Signal a bit needs to be discarded (`disc`).
- `01111110`: Flag the beginning/end of a frame (`flag`).
- `01111111...`: Error (7 or more 1s) (`err`).

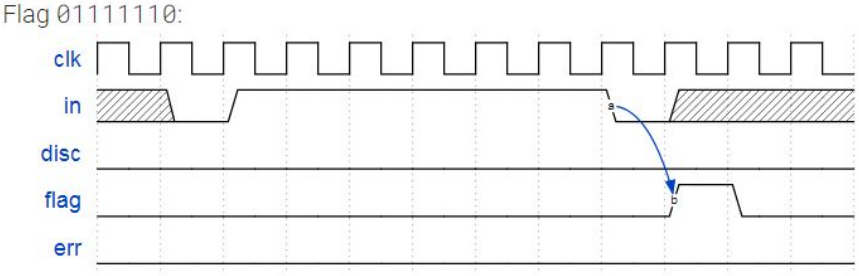
When the FSM is reset, it should be in a state that behaves as though the previous input were 0.

Here are some example sequences that illustrate the desired operation.

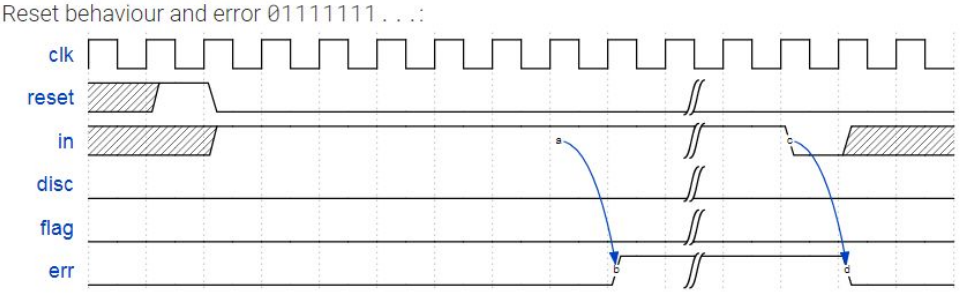
Discard `0111110`:



Flag `01111110`:



Reset behaviour and error `01111111...`:



Implement this state machine.

```
module top_module(
    input clk,
    input reset,    // Synchronous reset
    input in,
    output disc,
    output flag,
    output err);

    parameter S0=4'd0;
    parameter S1=4'd1;
    parameter S2=4'd2;
    parameter S3=4'd3;
    parameter S4=4'd4;
    parameter S5=4'd5;
    parameter S6=4'd6;
    parameter S7=4'd7; //more than 6 consecutive '1' = ERR
    parameter DISC =4'd8;
    parameter FLAG =4'd9;

    reg [3:0] curr_state;
    reg [3:0] next_state;

    always@(*)begin
        case(curr_state)
            S0:next_state=in?S1:S0;
            S1:next_state=in?S2:S0;
            S2:next_state=in?S3:S0;
            S3:next_state=in?S4:S0;
            S4:next_state=in?S5:S0;
            S5:next_state=in?S6:DISC;
            S6:next_state=in?S7:FLAG;
            S7:next_state=in?S7:S0;  //ERR
            DISC:next_state=in?S1:S0;
            FLAG:next_state=in?S1:S0;
            default:next_state=curr_state;
        endcase
    end

    always@(posedge clk)begin
        if(reset)begin
            curr_state<=4'd0;
        end
        else begin
            curr_state<=next_state;
        end
    end

    assign flag = curr_state==FLAG;
    assign disc = curr_state==DISC;
    assign err  = curr_state==S7;


endmodule
```