# Fsm ps2

The PS/2 mouse protocol sends messages that are three bytes long. However, within a continuous byte stream, it's not obvious where messages start and end. The only indication is that the first byte of each three byte message always has `bit[3]=1` (but bit[3] of the other two bytes may be 1 or 0 depending on data).

We want a finite state machine that will search for message boundaries when given an input byte stream. The algorithm we'll use is to discard bytes until we see one with `bit[3]=1`. We then assume that this is byte 1 of a message, and signal the receipt of a message once all 3 bytes have been received (done).
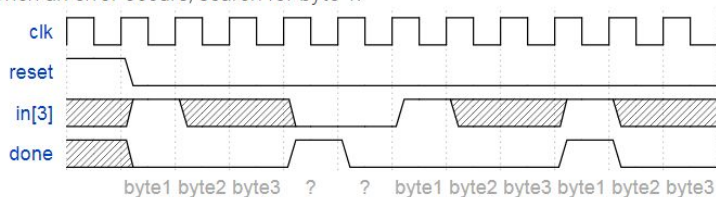
The FSM should signal done in the cycle immediately after the third byte of each message was successfully received.

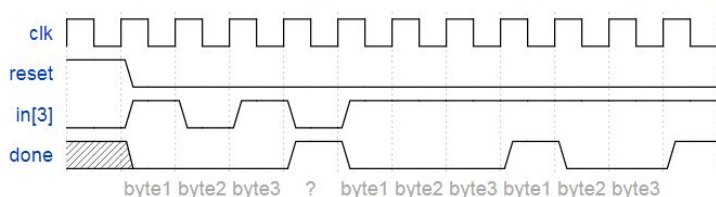## Some timing diagrams to explain the desired behaviour

Under error-free conditions, every three bytes form a message:
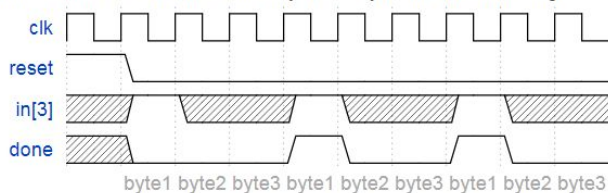
When an error occurs, search for byte 1:

Note that this is not the same as a `1xx` sequence recognizer. Overlapping sequences are not allowed here:


```verilog
module top_module(
    input clk,
    input [7:0] in,
    input reset,    // Synchronous reset
    output done); //

    reg [1:0] curr_state;
    reg [1:0] next_state;
```

```verilog
    parameter WAIT = 2'b00;
    parameter S1   = 2'b01;
    parameter S2   = 2'b10;
    parameter S3   = 2'b11;
    // State transition logic (combinational)
    always @(*) begin
        case(curr_state)
            WAIT: next_state = in[3]?S1:WAIT;
            S1  : next_state = S2;
            S2  : next_state = S3;
            S3  : next_state = in[3]?S1:WAIT;
        endcase
    end

    // State flip-flops (sequential)
    always @(posedge clk)begin
        if(reset)begin
            curr_state <= WAIT;
        end
        else begin
            curr_state <= next_state;
        end
    end

    // Output logic
    assign done = (curr_state == S3);

endmodule
```