```
module fsm (state, odd, even, terminal, pause, restart, clk, rst);

input pause, restart, clk, rst;
output [1:0] state;
output odd, even, terminal;
reg [1:0] state;
reg odd, even, terminal;
parameter [1:0] FIRST = 2'b11;
parameter [1:0] SECOND = 2'b01;
parameter [1:0] THIRD = 2'b10;

always_ff @(posedge clk or posedge rst) // sequential
begin
    if (rst) state <= FIRST;
    else
        begin
        case(state)
        FIRST: if (restart | pause) state <= FIRST;
                else state <= SECOND;
        SECOND: if (restart) state <= FIRST;
                else if (pause) state <= SECOND;
                else state <= THIRD;
        THIRD: if (!restart & pause) state <= THIRD;
                else state <= FIRST;
        default: state <= FIRST;
        endcase
        end
end
// output logic described using procedural assignment
always_comb
    begin
    odd = (state == FIRST) | (state == THIRD);
    even = (state == SECOND);
    terminal = (state == THIRD) & (restart | !pause);
    end
endmodule
```