module sequential\_circuit(

input wire clk,

input wire reset,

input wire M, // Motion sensor

input wire L, // Light sensor

input wire O, // Override

output reg light // Output light

);

// State encoding

typedef enum logic [2:0] {

S0 = 3'b000,

S1 = 3'b001,

S2 = 3'b010,

S3 = 3'b011,

S4 = 3'b100

} state\_t;

state\_t current\_state, next\_state;

// Sequential logic: state transition

always @(posedge clk or posedge reset) begin

if (reset)

current\_state <= S0; // Reset to initial state

else

current\_state <= next\_state;

end

// Combinational logic: next state and output logic

always @(\*) begin

// Default values

next\_state = current\_state;

light = 1'b0;

case (current\_state)

S0: begin

if (M && L == 1'b1 && O == 1'b0)

next\_state = S1;

else if (L == 1'b0 && O == 1'b0)

next\_state = S2;

end

S1: begin

if (M && L == 1'b0 && O == 1'b0)

next\_state = S3;

end

S2: begin

if (M == 1'b0 && L == 1'b0 && O == 1'b0)

next\_state = S2; // Remains in S2

end

S3: begin

light = 1'b1; // Light ON

if (M == 1'b0 && L == 1'b0 && O == 1'b0)

next\_state = S2;

end

S4: begin

light = 1'b1; // Light ON (override mode)

end

default: next\_state = S0;

endcase

// Override condition

if (O == 1'b1)

next\_state = S4;

end

endmodule