module security\_automation(

input wire clk, // System clock

input wire reset, // System reset

input wire motion\_sensor, // Motion sensor input

input wire door\_sensor, // Door sensor input

input wire window\_sensor, // Window sensor input

input wire arm\_system, // System arm switch

input wire disarm\_code\_entered, // Disarm code entry

output reg alarm // Alarm output

);

// State definitions

typedef enum logic [1:0] {

DISARMED = 2'b00,

ARMED = 2'b01,

ALERT = 2'b10

} state\_t;

state\_t current\_state, next\_state;

// Sequential logic for state transitions

always @(posedge clk or posedge reset) begin

if (reset) begin

current\_state <= DISARMED;

end else begin

current\_state <= next\_state;

end

end

// Combinational logic for next state

always @(\*) begin

case (current\_state)

DISARMED: begin

if (arm\_system) begin

next\_state = ARMED;

end else begin

next\_state = DISARMED;

end

end

ARMED: begin

if (motion\_sensor || door\_sensor || window\_sensor) begin

next\_state = ALERT; // Trigger alert if any sensor is triggered

end else if (!arm\_system) begin

next\_state = DISARMED; // Disarm if system is disarmed

end else begin

next\_state = ARMED;

end

end

ALERT: begin

if (disarm\_code\_entered) begin

next\_state = DISARMED; // Return to DISARMED if code is entered

end else begin

next\_state = ALERT;

end

end

default: next\_state = DISARMED;

endcase

end

// Output logic for alarm

always @(posedge clk or posedge reset) begin

if (reset) begin

alarm <= 1'b0;

end else begin

case (next\_state)

DISARMED: alarm <= 1'b0;

ARMED: alarm <= 1'b0;

ALERT: alarm <= 1'b1; // Alarm goes off in ALERT state

default: alarm <= 1'b0;

endcase

end

end

endmodule

// Testbench for the security\_automation module

module security\_automation\_tb;

reg clk;

reg reset;

reg motion\_sensor;

reg door\_sensor;

reg window\_sensor;

reg arm\_system;

reg disarm\_code\_entered;

wire alarm;

// Instantiate the module

security\_automation uut (

.clk(clk),

.reset(reset),

.motion\_sensor(motion\_sensor),

.door\_sensor(door\_sensor),

.window\_sensor(window\_sensor),

.arm\_system(arm\_system),

.disarm\_code\_entered(disarm\_code\_entered),

.alarm(alarm)

);

// Clock generation

initial begin

clk = 0;

forever #5 clk = ~clk; // 10ns clock period

end

// Test sequence

initial begin

// Initialize inputs

reset = 1;

motion\_sensor = 0;

door\_sensor = 0;

window\_sensor = 0;

arm\_system = 0;

disarm\_code\_entered = 0;

#10;

// Release reset

reset = 0;

#10;

// Arm the system

arm\_system = 1;

#20;

// Trigger motion sensor

motion\_sensor = 1;

#10;

motion\_sensor = 0;

// Enter disarm code

disarm\_code\_entered = 1;

#10;

disarm\_code\_entered = 0;

// Re-arm the system

arm\_system = 1;

#20;

// Trigger door sensor

door\_sensor = 1;

#10;

door\_sensor = 0;

// Enter disarm code

disarm\_code\_entered = 1;

#10;

disarm\_code\_entered = 0;

// End simulation

$stop;

end

endmodule