// Lighting Home Automation System Verilog Code

module lighting\_automation(

input wire clk, // System clock

input wire reset, // System reset

input wire motion\_sensor, // Motion sensor input

input wire light\_sensor, // Light sensor input (e.g., for daylight detection)

input wire manual\_override, // Manual override switch

output reg light // Output control for the light

);

// State definitions

typedef enum logic [1:0] {

OFF = 2'b00,

ON = 2'b01,

AUTO = 2'b10

} state\_t;

state\_t current\_state, next\_state;

// Parameters

parameter DARK\_THRESHOLD = 1'b0; // Light sensor threshold, e.g., 0 for dark

// Sequential block for state transitions

always @(posedge clk or posedge reset) begin

if (reset) begin

current\_state <= OFF;

end else begin

current\_state <= next\_state;

end

end

// Heating Home Automation System Verilog Code

module heating\_automation(

input wire clk, // System clock

input wire reset, // System reset

input wire [7:0] temp\_sensor, // Temperature sensor input (8-bit temperature value)

input wire manual\_override, // Manual override switch

input wire heating\_enable, // Enable heating system

output reg heater // Output control for the heater

);

// State definitions

typedef enum logic [1:0] {

OFF = 2'b00,

ON = 2'b01,

AUTO = 2'b10

} state\_t;

state\_t current\_state, next\_state;

// Parameters

parameter TEMP\_THRESHOLD = 8'd20; // Temperature threshold (e.g., 20 degrees)

// Sequential block for state transitions

always @(posedge clk or posedge reset) begin

if (reset) begin

current\_state <= OFF;

end else begin

current\_state <= next\_state;

end

end

// Combinational logic for next state

always @(\*) begin

case (current\_state)

OFF: begin

if (manual\_override || (heating\_enable && (temp\_sensor < TEMP\_THRESHOLD))) begin

next\_state = ON;

end else begin

next\_state = OFF;

end

end

ON: begin

if (!manual\_override && (temp\_sensor >= TEMP\_THRESHOLD)) begin

next\_state = OFF;

end else begin

next\_state = ON;

end

end

default: next\_state = OFF;

endcase

end

// Output logic

always @(posedge clk or posedge reset) begin

if (reset) begin

heater <= 1'b0;

end else begin

case (next\_state)

OFF: heater <= 1'b0;

ON: heater <= 1'b1;

default: heater <= 1'b0;

endcase

end

end

endmodule

// Home Security Automation System Verilog Code

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

// Sensors and Detectors Home Automation System Verilog Code

module sensors\_detectors\_automation(

input wire clk, // System clock

input wire reset, // System reset

input wire smoke\_sensor, // Smoke detector input

input wire gas\_sensor, // Gas leak detector input

input wire motion\_sensor, // Motion sensor input

input wire [7:0] temp\_sensor, // Temperature sensor input (8-bit temperature)

output reg smoke\_alert, // Smoke alert output

output reg gas\_alert, // Gas alert output

output reg motion\_alert, // Motion alert output

output reg temp\_alert // Temperature alert output

);

// Parameters

parameter TEMP\_THRESHOLD = 8'd30; // Temperature threshold (e.g., 30 degrees)

// Sequential block to monitor and update alerts based on sensor inputs

always @(posedge clk or posedge reset) begin

if (reset) begin

smoke\_alert <= 1'b0;

gas\_alert <= 1'b0;

motion\_alert <= 1'b0;

temp\_alert <= 1'b0;

end else begin

// Smoke Alert: Trigger when smoke sensor is activated

smoke\_alert <= smoke\_sensor;

// Gas Alert: Trigger when gas sensor is activated

gas\_alert <= gas\_sensor;

// Motion Alert: Trigger when motion is detected

motion\_alert <= motion\_sensor;

// Temperature Alert: Trigger if temperature exceeds threshold

temp\_alert <= (temp\_sensor > TEMP\_THRESHOLD) ? 1'b1 : 1'b0;

end

end

endmodule

// Smartphone/Tablet Controlled Home Automation System Verilog Code

module smartphone\_automation(

input wire clk, // System clock

input wire reset, // System reset

input wire [2:0] command, // Command from smartphone/tablet (3-bit for simplicity)

output reg light\_control, // Output control for lights

output reg heating\_control, // Output control for heating

output reg security\_control // Output control for security

);

// Command Definitions

parameter CMD\_LIGHT\_ON = 3'b001; // Command to turn on the light

parameter CMD\_LIGHT\_OFF = 3'b010; // Command to turn off the light

parameter CMD\_HEAT\_ON = 3'b011; // Command to turn on heating

parameter CMD\_HEAT\_OFF = 3'b100; // Command to turn off heating

parameter CMD\_SECURITY\_ON = 3'b101; // Command to arm security system

parameter CMD\_SECURITY\_OFF = 3'b110; // Command to disarm security system

// State Definitions

typedef enum logic [1:0] {

IDLE = 2'b00,

LIGHT = 2'b01,

HEATING = 2'b10,

SECURITY = 2'b11

} 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 <= IDLE;

end else begin

current\_state <= next\_state;

end

end

// Combinational Logic for Next State

always @(\*) begin

case (current\_state)

IDLE: begin

case (command)

CMD\_LIGHT\_ON, CMD\_LIGHT\_OFF: next\_state = LIGHT;

CMD\_HEAT\_ON, CMD\_HEAT\_OFF: next\_state = HEATING;

CMD\_SECURITY\_ON, CMD\_SECURITY\_OFF: next\_state = SECURITY;

default: next\_state = IDLE;

endcase

end

LIGHT: begin

if (command == CMD\_LIGHT\_ON || command == CMD\_LIGHT\_OFF) begin

next\_state = IDLE;

end else begin

next\_state = LIGHT;

end

end

HEATING: begin

if (command == CMD\_HEAT\_ON || command == CMD\_HEAT\_OFF) begin

next\_state = IDLE;

end else begin

next\_state = HEATING;

end

end

SECURITY: begin

if (command == CMD\_SECURITY\_ON || command == CMD\_SECURITY\_OFF) begin

next\_state = IDLE;

end else begin

next\_state = SECURITY;

end

end

default: next\_state = IDLE;

endcase

end

// Output Logic for Controlling Devices

always @(posedge clk or posedge reset) begin

if (reset) begin

light\_control <= 1'b0;

heating\_control <= 1'b0;

security\_control <= 1'b0;

end else begin

case (next\_state)

LIGHT: begin

if (command == CMD\_LIGHT\_ON) begin

light\_control <= 1'b1;

end else if (command == CMD\_LIGHT\_OFF) begin

light\_control <= 1'b0;

end

end

HEATING: begin

if (command == CMD\_HEAT\_ON) begin

heating\_control <= 1'b1;

end else if (command == CMD\_HEAT\_OFF) begin

heating\_control <= 1'b0;

end

end

SECURITY: begin

if (command == CMD\_SECURITY\_ON) begin

security\_control <= 1'b1;

end else if (command == CMD\_SECURITY\_OFF) begin

security\_control <= 1'b0;

end

end

default: begin

light\_control <= light\_control;

heating\_control <= heating\_control;

security\_control <= security\_control;

end

endcase

end

end

endmodule