module Lift8(clk, reset, req\_floor, idle, door, Up, Down, current\_floor, requests, max\_request, min\_request, emergency\_stop);

input clk, reset, emergency\_stop;

input [2:0] req\_floor; // 3-bit input for 8 floors (0 to 7)

output reg [1:0] door;

output reg [2:0] max\_request;

output reg [2:0] min\_request;

output reg [1:0] Up;

output reg [1:0] Down;

output reg [1:0] idle;

output reg [2:0] current\_floor;

output reg [7:0] requests;

reg door\_timer;

reg emergency\_stopped;

reg flag=0;

// Update requests when a new floor is requested

always @(req\_floor)

begin

requests[req\_floor] = 1;

// Update max\_request and min\_request based on requested floors

if (max\_request < req\_floor)

begin

max\_request = req\_floor;

end

if (min\_request > req\_floor)

begin

min\_request = req\_floor;

end

// Update max\_request and min\_request based on current floor

if (requests[max\_request] == 0 && req\_floor > current\_floor)

begin

max\_request = req\_floor;

end

if (requests[min\_request] == 0 && req\_floor < current\_floor)

begin

min\_request = req\_floor;

end

end

// Check and update lift behavior based on current floor

always @(current\_floor )

begin

if (requests[current\_floor] == 1)

begin

idle = 1;

door = 1;

requests[current\_floor] = 0;

door\_timer = 1; // Start the door timer when opening

end

end

// State machine for lift control

always @(posedge clk )

begin

if (door\_timer == 1)

begin

door <= 0; // Close the door after the one clock expires

//$display("%h", current\_floor);

end

if (reset)

begin

// Reset lift to initial state

flag=0;

current\_floor <= 0;

idle <= 0;

door <= 0; // door open

Up <= 1; // going up

Down <= 0; // not going down

max\_request <= 0;

min\_request <= 7;

requests <= 0;

emergency\_stopped <= 0; // Initialize emergency stop state

end

else if (requests == 0 && !reset)

begin

// Stay on the current floor if no requests

current\_floor <= current\_floor;

emergency\_stopped <= 0; // Clear emergency stop when not moving

end

// emergency

else if (emergency\_stop)

begin

// Emergency stop button is turned on

idle <= 1;

flag <=1;

emergency\_stopped <= 1; // Set emergency stop state

end

else if (emergency\_stopped && emergency\_stop)

begin

// Remain stopped until the emergency stop button is reset

current\_floor <= current\_floor;

door <= 0; // Keep the door closed during an emergency stop

end

// emergency reset

else if (!emergency\_stop && flag)

begin

// Emergency stop button is turned off

emergency\_stopped <= 0; // Set emergency stop state

flag <=0;

end

else

begin

// Normal operation when not in emergency stop

if (max\_request <= 7)

begin

if (min\_request < current\_floor && Down == 1)

begin

// Move down one floor

current\_floor <= current\_floor - 1;

door <= 0;

idle <= 0;

end

else if (max\_request > current\_floor && Up == 1)

begin

// Move up one floor

current\_floor <= current\_floor + 1;

door <= 0;

idle <= 0;

end

else if (req\_floor == current\_floor)

begin

// Open door and handle request

door <= 1;

idle <= 1;

end

else if (max\_request == current\_floor)

begin

Up <= 0;

Down <= 1;

end

else if (min\_request == current\_floor)

begin

Up <= 1;

Down <= 0;

end

end

end

end

endmodule

Complicated TB

module Lift8\_Tb();

reg clk, reset;

reg[2:0] req\_floor;

wire[1:0] idle, door, Up, Down;

wire[2:0] current\_floor;

wire [2:0] max\_request, min\_request;

wire [7:0] requests;

reg emergency\_stop;

Lift8 dut(

.clk(clk),

.reset(reset),

.req\_floor(req\_floor),

.idle(idle),

.door(door),

.Up(Up),

.Down(Down),

.current\_floor(current\_floor),

.max\_request(max\_request),

.min\_request(min\_request),

.requests(requests),

.emergency\_stop(emergency\_stop)

);

initial begin

$dumpfile("waveform.vcd"); // Specify the VCD waveform output file

$dumpvars(0, Lift8\_Tb); // Dump all variables in the module hierarchy

clk = 1'b0;

emergency\_stop = 0;

reset = 1;

#10;

reset = 0;

req\_floor = 1;

#30;

req\_floor = 4;

#10

// Simulate elevator operation

req\_floor = 3; // Request floor 3

#20;

req\_floor = 7; // Request floor 5

#20;

emergency\_stop = 1; // Activate emergency stop

#20;

emergency\_stop = 0; // Deactivate emergency stop

#10;

req\_floor = 2; // Request floor 2

#40;

req\_floor = 6; // Request floor 6

#20;

#20;

req\_floor = 1;

end

initial begin

$display("Starting simulation...");

$monitor("Time=%t,clk=%b,reset=%b,req\_floor=%h,idle=%h,door=%h,Up=%h,Down=%h,current\_floor=%h,max\_request=%h,min\_request=%h,requests=%h",

$time, clk, reset, req\_floor, idle, door, Up, Down, current\_floor, max\_request, min\_request, requests);

// Run the simulation for a sufficient duration

#305; // Adjust the simulation time as needed

$display("Simulation finished.");

$finish;

end

// C

always #5 clk = ~clk;

endmodule

Simple TB

module Lift8\_Tb();

reg clk, reset;

reg [2:0] req\_floor;

wire[1:0] idle, door, Up, Down;

wire [2:0] current\_floor;

wire[2:0] max\_request, min\_request;

wire [7:0] requests;

reg emergency\_stop;

Lift8 dut(

.clk(clk),

.reset(reset),

.req\_floor(req\_floor),

.idle(idle),

.door(door),

.Up(Up),

.Down(Down),

.current\_floor(current\_floor),

.max\_request(max\_request),

.min\_request(min\_request),

.requests(requests),

.emergency\_stop(emergency\_stop)

);

initial begin

$dumpfile("waveform.vcd"); // Specify the VCD waveform output file

$dumpvars(0, Lift8\_Tb); // Dump all variables in the module hierarchy

clk = 1'b0;

emergency\_stop = 0;

reset = 1;

#10;

reset = 0;

req\_floor = 1;

#30;

req\_floor = 4;

#40

// Simulate elevator operation

req\_floor = 7; // Request floor 7

#40;

req\_floor = 2; // Request floor 2

#50;

req\_floor = 6; // Request floor 6

#20;

req\_floor = 1;

end

initial begin

$display("Starting simulation...");

$monitor("Time=%t,clk=%b,reset=%b,req\_floor=%h,idle=%h,door=%h,Up=%h,Down=%h,current\_floor=%h,max\_request=%h,min\_request=%h,requests=%h",

$time, clk, reset, req\_floor, idle, door, Up, Down, current\_floor, max\_request, min\_request, requests);

// Run the simulation for a sufficient duration

#305; // Adjust the simulation time as needed

$display("Simulation finished.");

$finish;

end

// C

always #5 clk = ~clk;

endmodule