```
// ELEC402_PRJ1_SystemVerilog_FSM_Project

// Project Name: Food Food_Dispenser

// Name: Xingwei Su

// File: Food_Dispenser_FSM

// Description: Design's main logical FSM


module Food_Dispenser_FSM (

    input reset,                    //High when reset

    input clk,                      //Base clk input (fast)

    input timesup,                  //High when count time is up

    input [6:0] food_weight,        //Foot weight measure under plate

    input [6:0] set_food_weight,    //Initialized target food weight (set on UI interface module)

    input refill_detector,          //High when need refill

    input cap_detector,             //High when cap open

    input play_function_pedal,      //High when one full press cycle (press+release) is done

    input initialize_flag,          //High when UI interface module finish initialize

    input newday,                   //Posedge when new day


    output logic food_gate,         //High when gate open

    output logic warning,           //High when warning

    output logic play_function_flag,            //High when doing play function

    output logic play_function_fail_flag        //Posedge when fail

);


//Logic parameter setting

logic [3:0] state, nextstate;                       //15 states Max

logic [3:0] play_function_counter;                  //Counter counts to 15, daily play function limit 15 times


//State Parameters

parameter [3:0]     initialize_state                = 4'd0;

parameter [3:0]     initialize_wait_for_finish      = 4'd1;

parameter [3:0]     idle                            = 4'd2;


parameter [3:0]     before_open_gate                = 4'd3;

parameter [3:0]     add_food                        = 4'd4;

parameter [3:0]     weight_detect                   = 4'd5;

parameter [3:0]     add_food_complete               = 4'd6;
parameter [3:0]     wait_for_refill                  = 4'd7;

parameter [3:0]     play_function_detected          = 4'd8;
parameter [3:0]     play_function_fail              = 4'd9;
parameter [3:0]     play_function_complete          = 4'd10;

parameter [3:0]     newday_reset                    = 4'd11;



//Value parameter
parameter [6:0]     play_function_1g_food           = 7'd1;

//FSM flip-flop
always_ff @(posedge clk) begin
    if (reset)  state <= 3'd0;
    else        state <= nextstate;
end

//FSM
always_comb begin

    case (state)

        //Output values preset
        initialize_state: begin
            food_gate <= 1'b0;
            warning <= 1'b0;
            play_function_flag <= 1'b0;
            play_function_fail_flag <= 1'b0;
            play_function_counter <= 1'b0;
            nextstate <= initialize_wait_for_finish;
        end

        //Wait for UI initialization
        initialize_wait_for_finish:
            nextstate <= initialize_flag ? idle : initialize_wait_for_finish;
```

```
//Idle stage (detect flags)
idle:
    nextstate <= newday ? newday_reset :
                (refill_detector ? wait_for_refill :
                (timesup ? before_open_gate :
                (play_function_pedal ? play_function_detected : idle)));

//Detect if there is still food left on plate
//Warning if more than 1/3 target food weight is left
before_open_gate: begin
    warning <= (food_weight > (set_food_weight / 2'd3)) ? 1'b1 : 1'b0;
    nextstate <= add_food;
end

//Gate open
add_food:
    {food_gate, nextstate} <= {1'b1, weight_detect};

//Detect the weight of the food
//If on Play mode, only 1g of food will be send to plate
//If equal close gate
weight_detect:
    if (play_function_flag)
        nextstate <= (play_function_1g_food == food_weight) ? add_food_complete : weight_detect;
    else
        nextstate <= (set_food_weight == food_weight) ? add_food_complete : weight_detect;

//Gate close
add_food_complete: begin
    food_gate <= 1'b0;
    nextstate <= play_function_flag ? play_function_complete : idle;
end

//Refill food in bank
//Wait for cap to be closed to continue
wait_for_refill:        nextstate <= refill_detector ? wait_for_refill :
                                    (cap_detector ? wait_for_refill : idle);
        //Pedal trigered
        //Open function
        play_function_detected: begin
            play_function_fail_flag <= 1'b0;
            if (play_function_counter == 4'd15)
                nextstate <= play_function_fail;
            else
                {play_function_flag, nextstate} <= {1'b1, add_food};
        end

        //More than 15 times daily
        play_function_fail:     {nextstate, play_function_fail_flag} <= {idle, 1'b1};
            //blink LED (in module LED) <- test for posedge fail flag

        //Count up daily play times counter
        // Close function
        play_function_complete: begin
            play_function_counter <= play_function_counter + 1;
            play_function_flag <= 1'b0;
            nextstate <= idle;
        end

        newday_reset: begin
            play_function_counter <= 1'b0;
            nextstate <= idle;
        end

        //Default to initialization
        default:    nextstate <= initialize_state;
    endcase
end

endmodule
```

```
// ELEC402_PRJ1_SystemVerilog_FSM_Project
// Project Name: Food Food_Dispenser
// Name: Xingwei Su
// File: Food_Dispenser_TB
// Description: Testbench for module Food_Dispenser_FSM
// This program will test for reset, initialization, food refill, playground,
// auto feed functions for the design. Using .do file on ModelSim simulation
// for more detail.

module Food_Dispenser_TB();

    //Testbench simulate inputs
    logic           reset, clk, timesup, refill_detector, cap_detector, play_function_pedal, initialize_flag, newday;
    logic [6:0]     food_weight, set_food_weight;

    //Testbench simulate outputs
    logic           food_gate, warning, play_function_flag, play_function_fail_flag;


    Food_Dispenser_FSM DUT(
        .reset(reset),
        .clk(clk),
        .timesup(timesup),
        .food_weight(food_weight),
        .set_food_weight(set_food_weight),
        .refill_detector(refill_detector),
        .cap_detector(cap_detector),
        .play_function_pedal(play_function_pedal),
        .initialize_flag(initialize_flag),
        .newday(newday),

        .food_gate(food_gate),
        .warning(warning),
        .play_function_flag(play_function_flag),
        .play_function_fail_flag(play_function_fail_flag)
    );

    //Create a global fast clock
```

```
initial forever begin
    clk = 1;
    #10;
    clk = 0;
    #10;
end

initial begin

    timesup = 0;
    refill_detector = 0;
    cap_detector = 0;
    play_function_pedal = 0;
    initialize_flag = 0;
    food_weight = 0;
    set_food_weight = 7'd0;
    newday = 0;

    //reset stage
    reset = 1;
    #40;
    reset = 0;
    #40;



    //initialization stage complete
    set_food_weight = 7'd35;
    initialize_flag = 1'b1;
    #40;
    #40;

    //idle stage

    //refill needed
    refill_detector = 1'b1;
    #40;
    cap_detector = 1'b1;
    #40;
```

```
refill_detector = 1'b0;
#40;
cap_detector = 1'b0;
#40;
#40;

//Play_function Test
play_function_pedal = 1'b1;
#40;
play_function_pedal = 1'b0;
#40;
food_weight = 7'd1;
#40;
food_weight = 7'd0;
#40;
#40;
//test for fail flag
play_function_pedal = 1'b1;
food_weight = 7'd1;
#1800;
//test for new day (refreshing counter)
newday = 1'b1;
#40;
newday = 1'b0;
#120;
food_weight = 7'd0;
play_function_pedal = 1'b0;
#40;



//timesup & Warning test
timesup = 1'b1;
#40;
timesup = 1'b0;
#40;
```

```
        food_weight = 7'd12;

        #20;

        food_weight = 7'd35;

        #40;

        #40;

        food_weight = 7'd12;

        #40;

        timesup = 1'b1;

        #40;

        timesup = 1'b0;

        #40;

        food_weight = 7'd35;

        #40;

        food_weight = 7'd0;

        #40;

        timesup = 1'b1;

        #40;

        timesup = 1'b0;

        #40;

        food_weight = 7'd35;

        #40;




    end


endmodule
```